440EPx/GRx

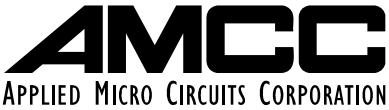
Preliminary User's Manual

PPC440EPx/GRX Embedded Processor

PPC440EPx/GRx Embedded Processor

User's Manual

PowerPC®



Applied Micro Circuits Corporation
6310 Sequence Drive, San Diego, CA 92121
Phone: (408) 450-9333 — (858) 535-6517 — Fax: (800) 840-6055
http://www.amcc.com (support@amcc.com)

AMCC reserves the right to make changes to its products, its data sheets, or related documentation, without notice and warrants its products solely pursuant to its terms and conditions of sale, only to substantially comply with the latest available data sheet. Please consult AMCC's Term and Conditions of Sale for its warranties and other terms, conditions and limitations. AMCC may discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information is current. AMCC does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others. AMCC reserves the right to ship devices of higher grade in place of those of lower grade.

AMCC SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

AMCC is a registered Trademark of Applied Micro Circuits Corporation. Copyright © 2007 Applied Micro Circuits Corporation. All Rights Reserved.

Contents

Figures 25

Tables 43

About This Book 49

Part I. Introduction 53

1. Overview 55
1.1 PPC440EPx and PPC440GRx Differences 55

2. On-Chip Buses 57
2.1 Processor Local Bus 57
2.1.1 PLB Features 57
2.1.2 PLB Master and Slave Assignments 58
2.1.3 PLB Master Priority Assignment 59
2.1.3.1 Alternate PLB4 Master Priority Register 0 (SDR0_AMP0) 60
2.1.3.2 Alternate PLB3 Master Priority Register 1 (SDR0_AMP1) 62
2.1.3.3 PPC440 CPU Control Register (SDR0_CP440) 63
2.1.3.4 PLB4 Master Interrupt Request Registers 0 and 1 (SDR0_MIRQ0, SDR0_MIRQ1) . 64
2.1.3.5 PLB Slave Address Pipeline Register (SDR0_SLPIPE0) 67
2.1.4 PLB4 Arbiter 68
2.1.4.1 PLB4 Arbiters 0 and 1 Registers 69
2.1.4.2 PLB4 Arbiter Revision ID Register (PLB4A0_REVID) 69
2.1.4.3 PLB4 Arbiter Control Register (PLB4An_ACR) 70
2.1.4.4 PLB4 Error Status Register Low (PLB4An_ESRL) 73
2.1.4.5 PLB4 Error Address Register Low (PLB4An_EARL) 75
2.1.4.6 PLB4 Error Address Register High (PLB4An_EARH) 75
2.1.4.7 PLB4 Crossbar Control Register (PLB4A0_CCR) 75
2.1.5 PLB4 to PLB3 Bridge Registers (Bridge Out 0) 76
2.1.5.1 PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL) 77
2.1.5.2 PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH) 77
2.1.5.3 PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0) 77
2.1.5.4 PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1) 80
2.1.5.5 PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG) 82
2.1.5.6 PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR) 83
2.1.5.7 PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR) 84
2.1.5.8 PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID) 84
2.1.6 PLB3 to PLB4 Bridge Registers (Bridge In 0) 84
2.1.6.1 PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL) 85
2.1.6.2 PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH) 85
2.1.6.3 PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0) 85
2.1.6.4 PLB3 to PLB4 Bridge Error Status Register 1 (P3P43BI0_BESR1) 87
2.1.6.5 PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG) 88
2.1.6.6 PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR) 89
2.1.6.7 PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR) 90
2.1.6.8 PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID) 90
2.1.7 PLB4 to OPB Bridge Registers 90
2.1.7.1 PLB4 to OPB Bridge Error Status Register 0 (PLB42OPBx_BESR0) 91
2.1.7.2 PLB4 to OPB Bridge Error Address Register Low (PLB42OPBx_BEARL) 93
2.1.7.3 PLB4 to OPB Bridge Error Address Register High (PLB42OPBx_BEARH) 93

2.1.7.4 PLB4 to OPB Bridge Error Status Register 1(PLB42OPBx_BESR1) 94
2.1.7.5 PLB4 to OPB Bridge Configuration Register (PLB42OPBx_CFG) 95
2.1.7.6 PLB4 to OPB Bridge Burst Latency Timer (PLB42OPBx_LATENCY) 95
2.1.7.7 PLB4 to OPB Bridge Revision ID (PLB42OPBx_REVID) 96
2.1.8 PLB3 Arbiter Registers 96
2.1.8.1 PLB3 Arbiter Revision ID Register (PLB3A0_REVID) 96
2.1.8.2 PLB3 Error Status Register (PLB3A0_BESR) 97
2.1.8.3 PLB3 Error Address Register (PLB3A0_BEAR) 98
2.1.8.4 PLB3 Arbiter 0 Control Register (PLB3A0_ACR) 98
2.1.9 PLB3 to OPB Bridge Registers 99
2.1.9.1 PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0) 100
2.1.9.2 PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR) 101
2.1.9.3 PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID) 101
2.1.9.4 PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1) 102
2.2 On-Chip Peripheral Bus 102
2.2.1 OPB Features 102
2.2.2 OPB Master Assignments 103
2.2.3 OPB Arbiter Registers 103
2.2.3.1 OPB Arbiter Priority Register (OPBAx_PR) 103
2.2.3.2 OPB Arbiter Control Register (OPBAx_CR) 104
2.2.4 OPB-to-PLB4 Bridge Registers 105
2.2.4.1 OPB-to-PLB4 Bridge Control Register (OPB2PLB40_BCTRL) 105
2.2.4.2 OPB-to-PLB4 Bridge Status Register (OPB2PLB40_BSTAT) 105
2.2.4.3 OPB-to-PLB4 Bridge Error Address Register Low (OPB2PLB40_BEARL) 106
2.2.4.4 OPB-to-PLB4 Bridge Error Address Register High (OPB2PLB40_BEARH) 106
2.2.4.5 OPB-to-PLB4 Bridge Revision ID Register (OPB2PLB40_REVID) 106
2.3 Device Control Register (DCR) Bus 107

Part II. PPC440EPx/GRx RISC Processor 109

3. Programming Model 111
3.1 Storage Addressing 111
3.2 User and Privileged Programming Models 111
3.2.1 Device Control Registers 115
3.2.2 Memory Mapped Registers 125
3.3 Instruction Classes 138

4. FPU Programming Model 139
4.1 Storage Addressing 139
4.1.1 Storage Operands 139
4.1.2 Effective Address Calculation 140
4.1.3 Data Storage Addressing Modes 140
4.2 Floating-Point Exceptions 141
4.3 Floating-Point Registers 141
4.3.1 Register Types 142
4.3.1.1 Floating-Point Registers (FPR0:FPR31) 142
4.3.1.2 Floating-Point Status and Control Register (FPSCR) 142
4.4 Floating-Point Data Formats 145
4.4.1 Value Representation 146
4.4.2 Binary Floating-Point Numbers 147
4.4.2.1 Normalized Numbers 147
4.4.2.2 Denormalized Numbers 147
4.4.2.3 Zero Values 147

4.4.3 Infinities 148
4.4.3.1 Not a Numbers 148
4.4.4 Sign of Result 149
4.4.5 Normalization and Denormalization 149
4.4.6 Data Handling and Precision 149
4.4.7 Rounding 150
4.5 Floating-Point Execution Models 151
4.5.1 Execution Model for IEEE Operations 152
4.5.2 Execution Model for Multiply-Add Type Instructions 154
4.6 Floating-Point Instructions 155
4.6.1 Instructions By Category 155
4.6.2 Load and Store Instructions 156
4.6.3 Floating-Point Store Instructions 157
4.6.4 Floating-Point Move Instructions 159
4.6.5 Floating-Point Arithmetic Instructions 159
4.6.5.1 Floating-Point Multiply-Add Instructions 159
4.6.6 Floating-Point Rounding and Conversion Instructions 160
4.6.7 Floating-Point Compare Instructions 160
4.6.8 Floating-Point Status and Control Register Instructions 161

5. Instruction and Data Caches 163

6. Memory Management 165

Part III. PPC440EPx/GRx System Operations 167

7. Reset and Initialization 169
7.1 Reset Signals 169
7.2 Reset Types 169
7.2.1 CPU Reset 169
7.2.2 Chip Reset 169
7.2.3 System Reset 170
7.3 Processor Core State After Reset 171
7.4 PPC440EPx/GRx Chip Initialization 175
7.4.1 Initial Configuration Register (CPR0_ICFG) 176
7.4.2 System Device Control Registers 176
7.4.2.1 Electronic Chip ID Register 0 (SDR0_ECID0) 177
7.4.2.2 Electronic Chip ID Register 1 (SDR0_ECID1) 177
7.4.2.3 Electronic Chip ID Register 2 (SDR0_ECID2) 177
7.4.2.4 Electronic Chip ID Register 3 (SDR0_ECID3) 177
7.4.2.5 Pin Function Control Register 0 (SDR0_PFC0) 177
7.4.2.6 Pin Function Control Register 1 (SDR0_PFC1) 178
7.4.2.7 GPIO Multiplexing Register (SDR0_PFC4) 179
7.4.2.8 Slave Address Pipeline Register (SDR0_SLPIPE0) 179
7.4.2.9 Soft Reset Registers (SDR0_SRST0 and SDR0_SRST1) 179
7.4.2.10 Miscellaneous Function Register (SDR0_MFR) 181
7.4.2.11 DDR Configuration Register (SDR0_DDRCFG) 182
7.5 Initialization Software Requirements 182

8. Bootstrap Controller 187
8.1 Bootstrap Configuration 187
8.2 Bootstrap Options 189
8.3 IIC Bootstrap Operations 193

8.3.1 Performance 194
8.4 IIC Bootstrap Registers 196
8.4.1 Pin Strapping Register (SDR0_PINSTP) 197
8.4.2 Serial Device Controller Settings Register (SDR0_SDCS0) 197
8.4.3 Serial Device Strap Register 0 (SDR0_SDSTP0) 198
8.4.4 Serial Device Strap Register 1 (SDR0_SDSTP1) 200
8.4.5 Read-Only Equivalent of SDR0_CUST0 (SDR0_SDSTP2) 201
8.4.6 Read-Only Version of SDR0_CUST1 (SDR0_SDSTP3) 202
8.4.7 Customer Configuration Register 0 (SDR0_CUST0) 203
8.4.8 Custom Configuration Register 1 (SDR0_CUST1) 204
8.4.9 EBC Configuration Register (SDR0_EBC0) 204

9. Universal Interrupt Controller 205
9.1 UIC Overview 205
9.2 UIC Features 205
9.3 UIC Interrupt Assignments 206
9.4 Interrupt Programmability 209
9.5 UIC Registers 210
9.5.1 UIC0 Status Register (UIC0_SR) 210
9.5.2 UIC1 Status Register (UIC1_SR) 212
9.5.3 UIC2 Status Register (UIC2_SR) 214
9.5.4 UIC0 Enable Register (UIC0_ER) 215
9.5.5 UIC1 Enable Register (UIC1_ER) 218
9.5.6 UIC2 Enable Register (UIC2_ER) 220
9.5.7 UIC0 Critical Register (UIC0_CR) 221
9.5.8 UIC1 Critical Register (UIC1_CR) 223
9.5.9 UIC2 Critical Register (UIC2_CR) 225
9.5.10 UIC0 Polarity Register (UIC0_PR) 226
9.5.11 UIC1 Polarity Register (UIC1_PR) 229
9.5.12 UIC2 Polarity Register (UIC2_PR) 231
9.5.13 UIC0 Trigger Register (UIC0_TR) 232
9.5.14 UIC1 Trigger Register (UIC1_TR) 234
9.5.15 UIC2 Trigger Register (UIC2_TR) 237
9.5.16 UIC0 Masked Status Register (UIC0_MSR) 237
9.5.17 UIC1 Masked Status Register (UIC1_MSR) 240
9.5.18 UIC2 Masked Status Register (UIC2_MSR) 242
9.5.19 UIC0 Vector Configuration Register (UIC0_VCR) 243
9.5.20 UIC1 Vector Configuration Register (UIC1_VCR) 243
9.5.21 UIC0 Vector Register (UIC0_VR) 244
9.5.21.1 Using the Value in UIC0_VR as a Vector Address or Entry Table Lookup 244
9.5.21.2 Vector Generation Scenarios 245
9.5.22 UIC1 Vector Register (UIC1_VR) 245
9.5.22.1 Using the Value in UIC1_VR as a Vector Address or Entry Table Lookup 246
9.5.22.2 Vector Generation Scenarios 246

10. Interrupts and Exceptions 247

11. Floating Point Unit Interrupts and Exceptions 249
11.1 Floating-Point Exceptions 249
11.2 Exceptions List 250
11.3 Floating-Point Interrupts 252
11.3.1 Floating-Point Unavailable Interrupt 252
11.4 Floating-Point Exception Behavior 253

11.4.1 Invalid Operation Exception 253
11.4.1.1 Action 254
11.4.2 Zero Divide Exception 255
11.4.2.1 Action 255
11.4.3 Overflow Exception 255
11.4.3.1 Action 255
11.4.4 Underflow Exception 256
11.4.4.1 Action 256
11.4.5 Inexact Exception 257
11.4.5.1 Action 257
11.5 Exception Priorities for Floating-Point Load and Store Instructions 258
11.6 Exception Priorities for other Floating-Point Instructions 258
11.7 QNaN 259
11.8 Updating FPRs on Exceptions 259
11.9 Floating-Point Status and Control Register 259
11.10 Updating the CR 260
11.10.1 CR Fields 260
11.10.2 Updating CR Fields 260
11.10.3 Generation of QNaN Results 261

12. Timer Facilities 263

13. General Purpose Timers 265
13.1 GPT Features 265
13.2 Time Base Counter 265
13.3 Compare Timers 265
13.3.1 Compare Timer Interrupt 266
13.4 GPT Registers 266
13.4.1 GPT Time Base Counter Register (GPT0_TBC) 267
13.4.2 GPT Interrupt Mask Register (GPT0_IM) 268
13.4.3 GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC) 268
13.4.4 GPT Interrupt Enable Register (GPT0_IE) 269
13.4.5 GPT Compare Timer Registers (GPT0_COMP0:GPT0_COMP6) 270
13.4.6 GPT Compare Mask Registers (GPT0_MASK0:GPT0_MASK6) 270
13.4.7 GPT Down Count Timer (GPT0_DCT0) 270
13.4.8 GPT Down Count Timer Interrupt Status (GPT0_DCIS) 271

14. Clocking 273
14.1 System Clocking 274
14.1.1 Feedback Selection 275
14.1.2 VCO Frequency and 'M' Value for SYS PLL 275
14.1.3 SYS PLL TUNE Setting 277
14.1.4 IIC Bootstrap Controller Clocking 277
14.1.5 Clocks For Off Chip Use 277
14.1.6 SYS PLL Strapping 278
14.1.7 PLL Bypass (Emulation Mode) 278
14.1.8 CPU / PLB Frequency N:1 Setting 278
14.1.9 Choosing System Clock Ratios 279
14.1.10 System Clock Ratio Examples 279
14.1.10.1 PLL Local Feedback Example 279
14.1.10.2 CPU Feedback Example 280
14.1.10.3 PerClk Feedback Example 281
14.2 PCI Clocking 282

14.2.1 PCI Adapter Applications 283
14.3 Clocking Registers 284
14.3.1 Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR) 284
14.3.2 Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA) 284
14.3.3 Clocking Update Register (CPR0_CLKUPD) 285
14.3.4 PLL Control Register (CPR0_PLLC0) 285
14.3.5 PLL Divisor Register (CPR0_PLLD0) 286
14.3.6 Primary A Divisor Register (CPR0_PRIMAD0) 287
14.3.7 Primary B Divisor Register (CPR0_PRIMBD0) 288
14.3.8 OPB Clock Divisor Register (CPR0_OPBD0) 288
14.3.9 Peripheral Clock Divisor Register (CPR0_PERD0) 288
14.3.10 MAL Clock Divisor Register (CPR0_MALD) 289
14.3.11 Sync PCI Clock Divisor Register (CPR0_SPCID) 289

15. Clock and Power Management 291
15.1 Overview 291
15.2 CPM Registers 291
15.2.1 CPM Enable Registers (CPM0_ER and CPM1_ER) 291
15.2.2 CPM Force Register (CPM0_FR and CPM1_FR) 293
15.2.3 CPM Status Register (CPM0_SR and CPM1_SR) 294

16. Debug Facilities 297

17. Security Function 299
17.1 Functional Description 299
17.1.1 Block Diagram 299
17.1.2 PLB Interface 300
17.1.3 Packet Engine 301
17.1.3.1 Input/Output Buffer 301
17.1.3.2 Controller 301
17.1.3.3 Header and Trailer Processor 301
17.1.3.4 Encryption/Decryption 302
17.1.3.5 Hash Function 303
17.1.3.6 Command Queue and Context Registers 303
17.1.3.7 Using the Packet Engine 303
17.1.4 DMA Controller 305
17.1.5 Public Key Accelerator and Public Key Memory 305
17.1.5.1 Operation 305
17.1.5.2 Functional Description 305
17.1.6 True Random Number Generator 305
17.1.7 Pseudo Random Number Generator 305
17.1.8 Interrupt Controller 306
17.1.9 Device Controller 306
17.2 Register Interface 306
17.2.1 Byte Ordering 306
17.2.2 Register Summary 306
17.2.3 CRYP0 SDR Control Register (SDR0_CRYP0) 312
17.2.4 PE Control/Status Register (CRYP0_PE_CTLST) 313
17.2.5 PE Source Address Register (CRYP0_PE_SOURCE) 319
17.2.6 PE Destination Address Register (CRYP0_PE_DEST) 319
17.2.7 PE SA Address Register (CRYP0_PE_SA) 320
17.2.8 PE Length Register (CRYP0_PE_LENGTH) 321
17.2.9 PE DMA Configuration Register (CRYP0_PE_DMA_CF) 322

17.2.10 PE DMA Status Register (CRYP0_PE_DMA_ST) 324
17.2.11 PE Packet Descriptor Ring Base Address Register (CRYP0_PE_PDR_BA) 326
17.2.12 PE Packet Result Ring Base Address Register (CRYP0_PE_RDR_BA) 327
17.2.13 PE Ring Size and Offset Register (CRYP0_PE_RING_S) 328
17.2.14 PE Ring Poll Register (CRYP0_PE_RING_P) 329
17.2.15 PE Internal Ring Status Register (CRYP0_PE_I_RING) 330
17.2.16 PE External Ring Status Register (CRYP0_PE_E_RING) 331
17.2.17 PE I/O Threshold Register (CRYP0_PE_IO_THR) 331
17.2.18 PE Gather Particle Ring Base Address Register (CRYP0_PE_GATH) 332
17.2.19 PE Scatter Particle Ring Base Address Register (CRYP0_PE_SCAT) 332
17.2.20 PE Particle Ring Size Register (CRYP0_PE_PT_S) 333
17.2.21 PE Particle Ring Configuration Register (CRYP0_PE_PT_CFG) 333
17.2.22 PE Particle Descriptor Source Address Register (CRYP0_PE_PR_SCA) 334
17.2.23 PE Particle Descriptor Source Control Register (CRYP0_PE_PR_SCC) 335
17.2.24 PE Particle Destination Address Register (CRYP0_PE_PR_DTA) 336
17.2.25 PE Particle Destination Control Register (CRYP0_PE_PR_DTC) 337
17.2.26 SA Command 0 Register (CRYP0_SA_CMD_0) 338
17.2.27 SA Command 1 Register (CRYP0_SA_CMD_1) 342
17.2.28 SA Key x Low/High Registers (CRYP0_SA_KEYx_L/H) 346
17.2.29 SA Inner Hash Digest x Registers (CRYP0_SA_IH_Dx) 347
17.2.30 SA Inner Outer Digest x Registers (CRYP0_SA_OH_Dx) 348
17.2.31 SA IPsec SPI Register (CRYP0_SA_SPI) 349
17.2.32 SA IPsec Sequence Number Register (CRYP0_SA_SEQ) 350
17.2.33 SA IPsec Sequence Number Mask High/Low Registers (CRYP0_SA_SEQMKH/L) 351
17.2.34 SA Nonce Value Register (CRYP0_SA_NONCE) 351
17.2.35 SA Pointer Register (CRYP0_SA_PNTR) 352
17.2.36 SA ARC4 i and j Pointer Register (CRYP0_SA_ARC4IJ) 352
17.2.37 SA ARC4 State Address Pointer Register (CRYP0_SA_ARC4SB) 353
17.2.38 SA Initialization Vector Registers (CRYP0_SA_IV_x) 353
17.2.39 SA Hash Byte Count Register (CRYP0_SA_HASH_B) 354
17.2.40 SA Inner Hash x (mirror of CRYP0_SA_IH_Dx) Registers (CRYP0_SA_IH_x) 354
17.2.41 SA ICV x—HMAC Result Registers (CRYP0_SA_ICV_x) 355
17.2.42 TRNG Output Register (CRYP0_TRNG_DATA) 356
17.2.43 TRNG Status Register (CRYP0_TRNG_STAT) 356
17.2.44 TRNG Control Register (CRYP0_TRNG_CTRL) 357
17.2.45 TRNG Entropy A Register (CRYP0_TRNG_ENTA) 358
17.2.46 TRNG Entropy B Register (CRYP0_TRNG_ENTB) 359
17.2.47 TRNG Test Seed x Registers (CRYP0_TRNG_Xx) 359
17.2.48 TRNG Counter Register (CRYP0_TRNG_CNTR) 360
17.2.49 TRNG Alarm Counter Register (CRYP0_TRNG_ALRM) 360
17.2.50 TRNG Configuration Register (CRYP0_TRNG_CFG) 361
17.2.51 TRNG Test Read of LFSR 0 Low/High Registers (CRYP0_TRNG_LF0L/H) 362
17.2.52 TRNG Test Read of LFSR 1 Low/High Registers (CRYP0_TRNG_LF1L/H) 363
17.2.53 TRNG Triple DES Key 0 Low/High Registers (CRYP0_TRNG_K0_L/H) 363
17.2.54 TRNG Triple DES Key 1 Low/High Registers (CRYP0_TRNG_K1_L/H) 364
17.2.55 TRNG Initialization Vector Low/High Registers (CRYP0_TRNG_IV_L/H) 364
17.2.56 PKA x Vector Address Register (CRYP0_PKA_x_PTR) 365
17.2.57 PKA x Vector Length Register (CRYP0_PKA_x_LEN) 365
17.2.58 PKA Shift Register (CRYP0_PKA_SHIFT) 366
17.2.59 PKA Function Code Register (CRYP0_PKA_FUNC) 367
17.2.60 PKA Comparison Result Register (CRYP0_PKA_COMP) 370
17.2.61 PKA Quotient MSW Register (CRYP0_PKA_DIV) 371
17.2.62 PKA Remainder MSW Register (CRYP0_PKA_MOD) 372

17.2.63 Interrupt Unmasked and Masked Status Registers (CRYP0_INT_UNMSK/MSK) 374
17.2.64 Interrupt Mask Register (CRYP0_INT_EN) 375
17.2.65 Interrupt Configuration Register (CRYP0_INT_CFG) 376
17.2.66 Interrupt Force Descriptor Read Register (CRYP0_INT_DESRD) 377
17.2.67 Interrupt Descriptor Count Register (CRYP0_INT_DESCT) 377
17.2.68 Device Control Register (CRYP0_DC_CTRL) 378
17.2.69 Device ID Register (CRYP0_DC_DEVID) 378
17.2.70 Device Information Register (CRYP0_DC_DEVINF) 379
17.2.71 DMA Source Address Register (CRYP0_DMA_USRC) 380
17.2.72 DMA Destination Address Register (CRYP0_DMA_UDST) 380
17.2.73 DMA Command Register (CRYP0_DMA_UCMD) 381
17.2.74 DMA Configuration/Status Register (CRYP0_DMA_CFG) 382
17.2.75 PRNG Status Register (CRYP0_PRNG_STAT) 383
17.2.76 PRNG Control Register (CRYP0_PRNG_CTRL) 384
17.2.77 PRNG Seed Value L/H Registers (CRYP0_PRNG_SDL/H) 385
17.2.78 PRNG Key x Low/High Registers (CRYP0_PRNG_KxL/H) 385
17.2.79 PRNG Result x Registers (CRYP0_PRNG_RSx) 386
17.2.80 PRNG LFSR Low/High Registers (CRYP0_PRNG_LFL/H) 386
17.3 Buffer Interface 387
17.3.1 ARC4 Buffer 387
17.3.2 Input buffer 387
17.3.3 Output Buffer 387
17.3.4 Public Key Memory (PKM) 388

18. KASUMI Algorithm 389
18.1 Features 389
18.2 References 389
18.3 Block Diagram 389
18.4 Functional Description 390
18.4.1 General Processing 390
18.4.1.1 KASUMI Mode 391
18.4.1.2 f8 Mode 391
18.4.1.3 f9 Mode 392
18.4.2 Flow Control 392
18.5 Register Interface 393
18.5.1 Byte Ordering 393
18.5.2 Data Input/Output Register (KASU0_DATAIN0, KASU0_DATAOUT0) 394
18.5.3 Data Input/Output Register (KASU0_DATAIN1, KASU0_DATAOUT1) 394
18.5.4 Control/Status Register (KASU0_CTRL, KASU0_STAT) 395
18.5.5 Mode Register (KASU0_MODE) 396
18.5.6 Key Registers (KASU0_KEYx) 397
18.5.7 Count Register (KASU0_COUNT) 398
18.5.8 Configuration Register (KASU0_CONFIG) 398
18.5.9 Fresh Register (KASU0_FRESH) 399
18.6 Programing Examples 399
18.6.1 KASUMI Mode Example 399
18.6.2 f8 Mode Example 400
18.6.3 f9 Mode Example 401

Part IV. PPC440EPx/GRx Peripheral Functions and Interfaces 403

19. DDR SDRAM Controller 405
19.1 Initialization Protocol 405

19.2 Device Address Mapping 406
19.2.1 DDR SDRAM Address Fields 406
19.2.1.1 Default Address Structure 406
19.2.1.2 Memory Mapping to Address Space 407
19.2.1.3 DDR Address Parameters for Common Organizations 409
19.3 Register Interface 410
19.3.1 DCR Access 410
19.3.2 SDRAM Device Control Registers 410
19.3.3 DCR Descriptions 412
19.4 Clocking 434
19.5 Error Checking and Correction (ECC) 435
19.5.1 Controlling ECC 435
19.5.2 ECC Error Types 436
19.5.3 ECC and Read Operations 437
19.5.4 ECC and Write Operations 437
19.5.5 Syndrome Codes 438
19.5.6 Forcing an ECC Error Event 438
19.5.7 Clearing a Reported ECC Event 439
19.6 Delay Compensation Circuit 439
19.7 Read Data Capture 440
19.8 Write Data Timing 444

20. Internal SRAM Controller 447
20.1 SRAM Controller Registers 447
20.1.1 SRAM Bank Configuration Register (SRAM0_SB0CR) 448
20.1.2 Bus Error Address Register (SRAM0_BEAR) 448
20.1.3 Bus Error Status Register 0 (SRAM0_BESR0) 449
20.1.4 Bus Error Status Register 1 (SRAM0_BESR1) 450
20.1.5 Power Management Register (SRAM0_PMEG) 452
20.1.6 Core ID Register (SRAM0_CID) 452
20.1.7 Revision ID Register (SRAM0_REVID) 453
20.1.8 Data Parity Checking Register (SRAM0_DPC) 453
20.2 Errors 453
20.2.1 Protection Error 453
20.2.1.1 BESR Field 453
20.2.2 Data Parity Error 454

21. Peripheral Component Interconnect (PCI) Interface 455
21.1 Features 455
21.2 Byte Ordering 456
21.3 PCI Bridge Functional Blocks 457
21.3.1 PLB-to-PCI Half-Bridge 457
21.3.2 PCI-to-PLB Half-Bridge 458
21.3.3 PCI Arbiter 458
21.4 PCI Bridge Address Mapping 459
21.4.1 PLB-to-PCI Address Mapping 459
21.4.2 PCI-to-PLB Address Mapping 461
21.4.3 PCI Target Map Configuration 462
21.5 PCI Bridge Transaction Handling 462
21.5.1 PLB-to-PCI Transaction Handling 463
21.5.1.1 PCI Master Commands 463
21.5.1.2 PLB Slave Read Handling 464

21.5.1.3 Prefetching 465
21.5.1.4 PLB Slave Write Handling 465
21.5.1.5 Aborted PLB Requests 465
21.5.1.6 Retried PCI Reads 466
21.5.2 PCI-to-PLB Transaction Handling 466
21.5.2.1 PLB Master Commands 467
21.5.2.2 Handling of Reads from PCI Masters 467
21.5.2.3 Miscellaneous 469
21.5.3 Completion Ordering 469
21.5.3.1 PCI Producer-Consumer Model 470
21.5.4 Collision Resolution 470
21.5.5 PCI Control Register (SDR0_PCI0) 470
21.6 PCI Bridge Configuration Registers 471
21.6.1 PCI Bridge Register Summary 471
21.6.2 PCI Bridge Local Configuration Registers 473
21.6.2.1 PMM 0 Local Address Register (PCIL0_PMM0LA) 473
21.6.2.2 PMM 0 Mask/Attribute Register (PCIL0_PMM0MA) 473
21.6.2.3 PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA) 474
21.6.2.4 PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA) 474
21.6.2.5 PMM 1 Local Address Register (PCIL0_PMM1LA) 474
21.6.2.6 PMM 1 Mask/Attribute Register (PCIL0_PMM1MA) 474
21.6.2.7 PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA) 475
21.6.2.8 PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA) 475
21.6.2.9 PMM 2 Local Address Register (PCIL0_PMM2LA) 475
21.6.2.10 PMM 2 Mask/Attribute Register (PCIL0_PMM2MA) 475
21.6.2.11 PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA) 476
21.6.2.12 PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA) 476
21.6.2.13 PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS) 476
21.6.2.14 PTM 1 Local Address Register (PCIL0_PTM1LA) 476
21.6.2.15 PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS) 477
21.6.2.16 PTM 2 Local Address Register (PCIL0_PTM2LA) 477
21.6.3 PCI Configuration Registers 477
21.6.3.1 PCI Configuration Address Register (PCIC0_CFGADDR) 478
21.6.3.2 PCI Configuration Data Register (PCIC0_CFGDATA) 478
21.6.3.3 PCI Vendor ID Register (PCIC0_VENDID) 479
21.6.3.4 PCI Device ID Register (PCIC0_DEVID) 479
21.6.3.5 PCI Command Register (PCIC0_CMD) 479
21.6.3.6 PCI Status Register (PCIC0_STATUS) 480
21.6.3.7 PCI Revision ID Register (PCIC0_REVID) 481
21.6.3.8 PCI Class Register (PCIC0_CLS) 481
21.6.3.9 PCI Cache Line Size Register (PCIC0_CACHELS) 482
21.6.3.10 PCI Latency Timer Register (PCIC0_LATTIM) 482
21.6.3.11 PCI Header Type Register (PCIC0_HDTYPE) 482
21.6.3.12 PCI Built-In Self Test (BIST) Control Register (PCIC0_BIST) 482
21.6.3.13 Unused PCI Base Address Register Space 483
21.6.3.14 PCI PTM 1 BAR (PCIC0_PTM1BAR) 483
21.6.3.15 PCI PTM 2 BAR (PCIC0_PTM2BAR) 483
21.6.3.16 PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID) 484
21.6.3.17 PCI Subsystem ID Register (PCIC0_SBSYSID) 484
21.6.3.18 PCI Capabilities Pointer (PCIC0_CAP) 484
21.6.3.19 PCI Interrupt Line Register (PCIC0_INTLN) 484
21.6.3.20 PCI Interrupt Pin Register (PCIC0_INTPN) 485
21.6.3.21 PCI Minimum Grant Register (PCIC0_MINGNT) 485

21.6.3.22 PCI Maximum Latency Register (PCIC0_MAXLTNCY) 485
21.6.3.23 PCI Interrupt Control/Status Register (PCIC0_ICS) 485
21.6.3.24 Error Enable Register (PCIC0_ERREN) 485
21.6.3.25 Error Status Register (PCIC0_ERRSTS) 486
21.6.3.26 Bridge Options 1 Register (PCIC0_BRDGOPT1) 487
21.6.3.27 PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0) 487
21.6.3.28 PLB Slave Error Syndrome Register 1 (PCIC0_PLBBESR1) 489
21.6.3.29 PLB Slave Error Address Register (PCIC0_PLBBEAR) 490
21.6.3.30 Capability Identifier (PCIC0_CAPID) 490
21.6.3.31 Next Item Pointer (PCIC0_NEXTIPTR) 490
21.6.3.32 Power Management Capabilities (PCIC0_PMC) 491
21.6.3.33 Power Management Control/Status Register (PCIC0_PMCSR) 491
21.6.3.34 PMCSR PCI-to-PCI Bridge Support Extensions (PCIC0_PMCSRBSE) 492
21.6.3.35 PCI Data Register (PCIC0_DATA) 492
21.6.3.36 Bridge Options 2 Register (PCIC0_BRDGOPT2) 492
21.6.3.37 Power Management State Change Request Register (PCIC0_PMSCRR) 493
21.7 Error Handling 493
21.7.1 PLB Unsupported Transfer Type 494
21.7.2 PCI Master Abort 494
21.7.3 Bridge PCI Master Receives Target Abort While PCI Bus Master 495
21.7.4 PCI Target Data Bus Parity Error Detection 495
21.7.5 PCI Master Data Bus Parity Error Detection 496
21.7.6 PCI Address Bus Parity Error While PCI Target 496
21.7.7 PLB Master Bus Error Detection 496
21.8 PCI Power Management Interface 497
21.8.1 Capabilities and Power Management Status and Control Registers 497
21.8.2 Power State Control 497
21.8.3 Changing Power States 497
21.9 PCI Bridge Reset and Initialization 498
21.9.1 Address Map Initialization 498
21.9.2 Other Configuration Register Initialization 500
21.9.3 Target Bridge Initialization 500
21.9.4 Local Processor Boot from PCI Memory 501
21.9.5 Type 0 Configuration Cycles for Other Devices 501
21.10 Timing Diagrams 501
21.10.1 PCI Timing Diagram Descriptions 502
21.10.1.1 PCI Master Burst Read From SDRAM 502
21.10.1.2 PCI Master Burst Write To SDRAM 502
21.10.1.3 CPU Read From PCI Memory Slave, Nonprefetching 502
21.10.1.4 CPU Read From PCI Memory Slave, Prefetching 502
21.10.1.5 CPU Write To PCI Memory Slave 502
21.10.1.6 PCI Memory To SDRAM DMA Transfer 503
21.10.1.7 SDRAM To PCI Memory DMA Transfer 503
21.10.2 Asynchronous 503
21.10.3 Synchronous 524

22. External Bus Controller 555

22.1 Interface Signals 555
22.1.1 Interfacing to Byte, Half-Word, and Word Devices 556
22.1.2 Driver Enables 557
22.2 Non-Burst Peripheral Bus Transactions 558
22.2.1 Single Read Transfer 558
22.2.2 Single Write Transfer 559

22.3 Burst Transactions 560
22.3.1 Burst Read Transfer 561
22.3.2 Burst Write Transfer 562
22.4 Device-Paced Transfers 563
22.4.1 Device-Paced Single Read Transfer 564
22.4.2 Device-Paced Single Write Transfer 565
22.4.3 Device-Paced Burst Read Transfer 566
22.4.4 Device-Paced Burst Write Transfer 567
22.5 External Bus Master Interface 568
22.5.1 Arbitration 569
22.5.2 Transaction Overview 570
22.5.3 Single Read and Single Write Transfers 571
22.5.4 Burst Read Transfer 572
22.5.5 Burst Write Transfer 573
22.5.6 External Master Error Interrupts 574
22.6 EBC Registers 574
22.6.1 EBC Configuration Register (EBC0_CFG) 576
22.6.2 Peripheral Bank Configuration Registers (EBC0_B0CR:EBC0_B5CR) 577
22.6.3 Peripheral Bank Access Parameters (EBC0_B0AP:B5AP) 578
22.7 Error Reporting 580
22.7.1 Peripheral Bus Error Address Register (EBC0_BEAR) 580
22.7.2 Peripheral Bus Error Status Register 0 (EBC0_BESR0) 580

23. NAND Flash Controller 583
23.1 Overview 584
23.2 NDFC Signal Multiplexing 585
23.3 NDFC Interface 585
23.4 Resetting the Controller 585
23.5 Configuring EBC to Support the NAND Flash Controller 585
23.5.1 EBC0_BxAP 586
23.5.2 EBC0_CFG 586
23.5.3 EBC0_BxCR 586
23.5.4 Summary of EBC and NDFC Interface Configurations 586
23.6 Booting from NDFC 587
23.6.1 AutoRead Mode 589
23.6.2 Accessing NAND Flash 591
23.6.2.1 Reading NAND Flash 591
23.6.2.2 Writing NAND Flash 591
23.7.1 Column Parity 594
23.7.2 Line/Row Parity 594
23.7.3 How to generate line parity (LP00-LP15) 595
23.8 External Device Cycles 596
23.8.1 Command Cycle 597
23.8.2 Address Write Cycle 597
23.8.3 Basic Nand Flash Device Read Cycle 598
23.8.4 Basic Nand Flash Device Write Cycle 599
23.9.1 Memory Map 600
23.9.2 NDFC Address Register (NDFC0_ADDR) 601
23.9.3 NDFC Command Register (NDFC0_CMD) 601
23.9.4 NDFC Data Register (NDFC0_DATA) 601
23.9.5 NAND Flash ECC Calculation Registers (NDFC0_ECC0:NDFC0_ECC7) 603
23.9.6 NDFC Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR) 603

23.9.7 NDFC Configuration Register (NDFC0_CR) 604
23.9.8 NDFC Status Register (NDFC0_SR) 606
23.9.9 NAND Flash Direct Hardware Control Register (NDFC0_HWCTL) 607
23.9.10 NDFC Revision ID Register (NDFC0_REVID) 608

24. Direct Memory Access Controllers 609
24.1 DMA to PLB4 Controller (DMA2P40) 609
24.1.1 DMA Transfers 609
24.1.1.1 Memory-to-Memory Transfers 610
24.1.1.2 Scatter/Gather Transfers 611
24.1.2 Configuration and Status Registers 611
24.1.2.1 DMA to PLB 4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3) 613
24.1.2.2 DMA to PLB 4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3)) 614
24.1.2.3 DMA to PLB 4 Source Address Registers 615
24.1.2.4 DMA to PLB 4 Destination Address Registers 616
24.1.2.5 DMA to PLB 4 Scatter/Gather Descriptor Address Registers 616
24.1.2.6 DMA to PLB 4 Status Register (DMA2P40_SR) 617
24.1.2.7 DMA to PLB 4 Scatter/Gather Command Register (DMA2P40_SGC) 618
24.1.2.8 DMA to PLB 4 Sleep Mode Register (DMA2P40_SLP) 618
24.1.2.9 DMA to PLB 4 Polarity Configuration Register (DMA2P40_POL) 619
24.1.3 Channel Priorities 619
24.1.4 Data Parity During DMA Peripheral Transfers 619
24.1.5 Peripheral and Device-Paced Memory Bursts 619
24.1.6 Errors 620
24.1.6.1 Address Alignment Error 620
24.1.6.2 Burst Count Error 620
24.1.6.3 Burst Prefetch Error 621
24.1.6.4 PLB or OPB Timeout 621
24.1.6.5 Slave Transfer Errors 621
24.1.7 DMA to PLB4 Interrupts 621
24.1.8 Scatter/Gather Transfers 622
24.1.9 Programming the DMA2P40 Controller 623
24.1.9.1 Memory-to-Memory Transfers 623
24.2 DMA to PLB3 Controller (DMA2P30) 625
24.2.1 External Interface Signals 626
24.2.2 Functional Overview 627
24.2.3 Peripheral Mode Transfers 627
24.2.4 Memory-to-Memory Transfers 628
24.2.5 Scatter/Gather Transfers 628
24.2.6 Configuration and Status Registers 629
24.2.6.1 DMA to PLB 3 Polarity Configuration Register (DMA2P30_POL) 630
24.2.6.2 DMA to PLB 3 Sleep Mode Register (DMA2P30_SLP) 631
24.2.6.3 DMA to PLB 3 Status Register (DMA2P30_SR) 631
24.2.6.4 DMA to PLB 3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3) 632
24.2.6.5 DMA to PLB 3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3) 634
24.2.6.6 DMA to PLB 3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3) 634
24.2.6.7 DMA to PLB 3 Count Registers (DMA2P30_CT0–DMA2P30_CT3) 634
24.2.6.8 DMA to PLB 3 Scatter/Gather Descriptor Address Registers (DMA2P30_SG0–DMA2P30_SG3) 635
24.2.6.9 DMA to PLB 3 Scatter/Gather Command Register (DMA2P30_SGC) 635
24.2.6.10 DMA to PLB 3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3) 636
24.2.6.11 DMA to PLB 3 Address Decode Register (DMA2P30_ADR) 636
24.2.7 Channel Priorities 636

24.2.8 Data Parity During DMA Peripheral Transfers 637
24.2.9 Errors 637
24.2.10 Address Alignment Error 637
24.2.11 PLB Timeout 638
24.2.12 Slave Errors 638
24.2.13 DMA Interrupts 638
24.2.14 Scatter/Gather Transfers 638
24.2.15 Programming the DMA to PLB3 Controller 640
24.2.16 Peripheral Mode Transfers 640
24.2.16.1 Peripheral-to-Memory Transfer 642
24.2.16.2 Memory-to-Peripheral Transfer 643
24.2.17 Memory-to-Memory Transfers 643
24.2.17.1 Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers 643
24.2.17.2 Software-Initiated Memory-to-Memory Transfers (Non-Device Paced) 644

25. Memory Access Layer 645
25.1 MAL Features 645
25.1.1 MAL Internal Structure 647
25.1.1.1 PLB Master 647
25.1.1.2 EOPB Master 647
25.1.1.3 TX Channel Handler 647
25.1.1.4 RX Channel Handler 647
25.1.1.5 TX Channel Arbiter 648
25.1.1.6 RX Channel Arbiter 648
25.1.1.7 TX Common Channel Logic 648
25.1.1.8 RX Common Channel Logic 648
25.1.1.9 Register Map File 648
25.2 MAL0 Interfaces and Channel Assignments 648
25.3 Transmit and Receive Operations 649
25.3.1 Transmit Operations 649
25.3.2 Receive Operations 650
25.3.3 PPC440EPx/GRx System Device Control Registers 651
25.3.3.1 MAL Receive Burst Length Register (SDR0_MALRBL) 651
25.3.3.2 MAL Receive Bus Size Register (SDR0_MALRBS) 651
25.3.3.3 MAL Transmit Burst Length Register (SDR0_MALTBL) 652
25.3.3.4 MAL Transmit Bus Size Register (SDR0_MALTBS) 652
25.4 Buffer Descriptor 652
25.5 Transmit Software Interface 654
25.5.1 Wrapping the BD Table for Transmit 655
25.5.2 Continuous Mode for Transmit 655
25.5.3 Back Up a Packet for Transmit 655
25.5.4 Descriptor Not Valid for Transmit 656
25.5.5 Scroll Descriptors for Transmit 656
25.6 Receive Software Interface 657
25.6.1 Wrapping the BD Table for Receive 657
25.6.2 Continuous Mode for Receive 657
25.6.3 Descriptor Not Valid for Receive 657
25.6.4 Buffer Length for Receive 658
25.7 Buffer Descriptor Status/Control Fields 658
25.7.1 Information from a Software Device Driver Directed To MAL and EMAC 658
25.7.2 Information from MAL and EMAC Directed to Software 658
25.7.3 Status/Control Field Handling 658
25.7.4 Status/Control Field Format 659

25.7.5 TX Status/Control Field Format 660
25.7.6 RX Status/Control Field Format 661
25.8 MAL Programming Notes 662
25.8.1 MAL Initialization 662
25.8.2 Interrupts 662
25.8.3 Error Handling 663
25.8.3.1 Error Detection 663
25.8.3.2 Indicated Errors 663
25.8.3.3 Error Handling Registers 664
25.8.3.4 Operational Error Modes 665
25.8.3.5 Resolution of an Error Situation 665
25.8.3.6 Interrupts To Software 666
25.9 MAL Registers 668
25.9.1 MAL Configuration Register (MAL0_CFG) 669
25.9.2 Channel Active Set and Reset Registers 670
25.9.3 End of Buffer Interrupt Status Registers 671
25.10 Error Registers 672
25.10.1 MAL Error Status Register (MAL0_ESR) 672
25.10.2 MAL Interrupt Enable Register (MAL0_IER) 674
25.10.3 Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR) 674
25.10.4 PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR) 675
25.10.5 Channel Table Pointer Registers 675
25.10.6 Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR) 676
25.10.7 Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR) 676
25.10.8 RX Channel Buffer Size Register (MAL0_RCBSx) 677
25.11 Interrupt Coalescence 677
25.11.1 Coalescing by Frame Counter Method 678
25.11.2 Coalescing by Timer Threshold 678
25.12 Interrupt Coalescence Registers 678
25.12.1 Interrupt Coalescing Control Transmit Register (SDR0_ICCRTX) 678
25.12.2 Interrupt Coalescing Control Receive Register (SDR0_ICCRRX) 679
25.12.3 Interrupt Coalescing Timer Register Transmit Channels 0 and 1 (SDR0_ICTRTXx) 680
25.12.4 Interrupt Coalescing Timer Register Receive Channel 0 and 1 (SDR0_ICTRRXx) 680
25.12.5 Interrupt Coalescing Status Register Transmit Channel 0 and 1 (SDR0_ICSRTXx) 680
25.12.6 Interrupt Coalescing Status Register Receive Channel 0 and 1 (SDR0_ICSRRXx) 680

26. EMAC to PHY Interface Bridges 681
26.1 ZMII Bridge 682
26.2 ZMII Features 683
26.3 ZMII Bridge Interface Signals 683
26.4 EMAC-ZMII Bridge Interfaces 683
26.4.1 MII Interface 683
26.4.1.1 SMII Interface 684
26.5 ZMII Bridge Registers 684
26.5.1 Function Enable Register (ZMII0_FER) 685
26.5.2 Speed Select Register (ZMII0_SSR) 685
26.5.3 SMII Status Register (ZMII0_SMIISR) 686
26.6 RGMII Bridge 687
26.6.1 RGMII Bridge Features 687
26.6.2 RGMII Bridge Interface 687
26.6.3 Ethernet Signal Grouping for Ethernet to PHY Bridges 689
26.6.4 EMAC-RGMII Bridge Interfaces 689

26.6.5 RGMII Bridge Registers 689
26.6.5.1 RGMII Function Enable Register (RGMII0_FER) 690
26.6.5.2 Speed Selection Register (RGMII0_SSR) 690
26.7 Phase-Locked Loop (PLL) Dedicated to the RGMII Bridge 691
26.7.1 Ethernet PLL Configuration Register (SDR0_PFC2) 691

27. Ethernet Media Access Controllers 693
27.1 EMAC Features 695
27.2 EMAC Operation 696
27.2.1 MAL Slave Logic 696
27.2.2 OPB Slave Logic 697
27.2.3 FIFO Management Logic 697
27.2.4 Ethernet Address Match Logic 697
27.2.5 Configuration and Status Registers 697
27.2.6 Wake On LAN Logic 697
27.2.7 Ethernet MAC 697
27.2.8 EMAC Loopback Modes 698
27.2.9 Packet Rejection 698
27.2.9.1 EMACx RX Status Register 699
27.2.9.2 EMACx TX Status Register 701
27.2.9.3 EMACx RX Packet Reject Counter 702
27.3 EMAC Transmit Operation 703
27.3.1 Arbitration Between TX Channels 703
27.3.1.1 MAL TX Descriptor Control/Status Field 703
27.3.1.2 Early Packet Termination during Transmit 705
27.3.1.3 Empty Packets 705
27.3.1.4 Automatic Retransmission of Colliding Packets 705
27.3.1.5 Inter-Packet Gap (IPG) Tuning 705
27.3.1.6 Full-Duplex Operation 706
27.3.1.7 Packet Content Configuration Options 706
27.4 EMAC Receive Operation 709
27.4.1 EMAC – MAL RX Packet Transfer Flow 709
27.4.2 MAL RX Descriptor Status 709
27.4.3 Early Packet Termination during Receive 710
27.4.4 Discarding Packets During Receive 710
27.4.5 Wakeup On Lan (WOL) Support 711
27.4.5.1 EMAC WOL Support 711
27.5 Flow Control 712
27.5.1 MAC Control Packet 712
27.5.2 Control Packet Transmission 713
27.5.3 Integrated Flow Control 713
27.5.4 Control Packet Reception 714
27.6 VLAN Support 715
27.6.1 VLAN Tagged Packet Transmission 716
27.6.2 VLAN Tagged Packet Reception 717
27.6.3 Address Match Mechanism 717
27.6.3.1 Non-WOL Mode 717
27.6.3.2 WOL Mode 719
27.7 EMAC Registers 720
27.7.1 Mode Register 0 (EMACx_MR0) 723
27.7.2 Mode Register 1 (EMACx_MR1) 723
27.7.3 Transmit Mode Register 0 (EMACx_TMR0) 725

27.7.4 Transmit Mode Register 1 (EMACx_TMR1) 726
27.7.4.1 Low-Priority Requests 726
27.7.4.2 Urgent-Priority Requests 726
27.7.5 Receive Mode Register (EMACx_RMR) 726
27.7.6 Interrupt Status Register (EMACx_ISR) 728
27.7.7 Interrupt Status Enable Register (EMACx_ISER) 730
27.7.8 Individual Address High (EMACx_IAHR) 732
27.7.9 Individual Address Low (EMACx_IALR) 732
27.7.10 VLAN TPID Register (EMACx_VTPID) 733
27.7.11 VLAN TCI Register (EMACx_VTCI) 733
27.7.12 Pause Timer Register (EMACx_PTR) 733
27.7.13 Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4) 734
27.7.14 Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4) 734
27.7.15 Last Source Address High (EMACx_LSAH) 734
27.7.16 Last Source Address Low (EMACx_LSAL) 734
27.7.17 Inter-Packet Gap Value Register (EMACx_IPGVR) 735
27.7.18 STA Control Register (EMACx_STACR) 735
27.7.19 Transmit Request Threshold Register (EMACx_TRTR) 736
27.7.20 Receive Low/High Water Mark Register (EMACx_RWMR) 737
27.7.21 Transmitted Octects (EMACx_OCTX) 738
27.7.22 Received Octects Register (EMACx_OCRX) 738
27.8 MII/GMII Interface 738
27.8.1 MII Station Management Interface 738
27.9 MAL – EMAC Packet Transfer Flow 739
27.10 Programming Notes 739
27.10.1 Reset and Initialization 740
27.10.1.1 Scenario 1 740
27.10.1.2 Scenario 2 741
27.10.1.3 Scenario 3 741

28. Serial Port Operations 743
28.1 Functional Description 743
28.2 Serial Port Clocking 744
28.3 UART Registers 746
28.3.1 Receiver Buffer Registers (UARTx_RBR) 747
28.3.2 Transmitter Holding Registers (UARTx_THR) 747
28.3.3 Interrupt Enable Registers (UARTx_IER) 747
28.3.4 Interrupt Identification Registers (UARTx_IIR) 748
28.3.5 FIFO Control Registers (UARTx_FCR) 749
28.3.6 Line Control Registers (UARTx_LCR) 749
28.3.7 Modem Control Registers (UARTx_MCR) 750
28.3.8 Line Status Registers (UARTx_LSR) 751
28.3.9 Modem Status Registers (UARTx_MSR) 752
28.3.10 Scratchpad Registers (UARTx_SCR) 753
28.3.11 Divisor Latch LSB and MSB Registers (UARTx_DLL and UARTx_DLM) 753
28.3.12 UART Configuration Register 0 (SDR0_UART0) 754
28.3.13 UART Configuration Register 1 (SDR0_UART1) 755
28.3.14 UART Configuration Register 2 (SDR0_UART2) 756
28.3.15 UART Configuration Register 3 (SDR0_UART3) 756
28.4 FIFO Operation 757
28.4.1 Interrupt Mode 757
28.4.1.1 Receiver 757
28.4.1.2 Transmitter 758

28.4.2 Polled Mode 758
28.4.3 UART and Sleep Mode 758
28.5 DMA Operation 759
28.5.1 Transmitter DMA Mode 760
28.5.2 Receiver DMA Mode 764

29. Serial Peripheral Interface 769
29.1 Functional Description 769
29.1.1 SPI Interrupt Operation 770
29.1.2 Loopback 770
29.1.3 Typical Operation (Exchange of Data) 771
29.2 SPI Registers 771
29.2.1 SPI Receive Data Register (SPI0_RxD) 772
29.2.2 SPI Transmit Data Register (SPI0_TxD) 772
29.2.3 SPI Control Register (SPI0_CR) 772
29.2.4 SPI Clock Divisor Modulus Register (SPI0_CDM) 773
29.2.5 SPI Status Register (SPI0_SR) 773
29.2.6 SPI Mode Register (SPI0_MODE) 774

30. IIC Bus Interface 775
30.1 Addressing 775
30.1.1 Addressing Modes 775
30.1.2 Seven-Bit Addresses 775
30.1.3 Ten-Bit Addresses 776
30.2 IIC Registers 777
30.3 IIC Register Descriptions 778
30.3.1 IICx Master Data Buffer (IICx_MDBUF) 778
30.3.2 IICx Slave Data Buffer (IICx_SDBUF) 778
30.3.3 IICx Low Master Address Register (IICx_LMADR) 779
30.3.4 IICx High Master Address Register (IICx_HMADR) 780
30.3.5 IICx Control Register (IICx_CNTL) 780
30.3.6 IICx Mode Control Register (IICx_MDCNTL) 781
30.3.7 IICx Status Register (IICx_STS) 782
30.3.8 IICx Extended Status Register (IICx_EXTSTS) 784
30.3.9 IICx Low Slave Address Register (IICx_LSADR) 785
30.3.10 IICx High Slave Address Register (IICx_HSADR) 786
30.3.11 IICx Clock Divide Register (IICx_CLKDIV) 786
30.3.12 IICx Interrupt Mask Register (IICx_INTRMSK) 787
30.3.13 IICx Transfer Count Register (IICx_XFRCNT) 788
30.3.14 IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) 789
30.3.15 IICx Direct Control Register (IICx_DIRECTCNTL) 790
30.3.16 IICx Interrupt Register (IICx_INTR) 791
30.4 Interrupt Handling 792
30.5 General Considerations 792

31. GPIO Operations 795
31.1 Overview 795
31.2 Features 797
31.3 Clock and Power Management 797
31.4 GPIO Registers 797
31.5 GPIO Register Reset Values 799
31.6 Detailed Register Descriptions 799

31.6.1 GPIO Output Register (GPIOx_OR) 799
31.6.2 GPIO Three-State Control Register (GPIOx_TCR) 799
31.6.3 GPIO Output Select Registers (GPIOx_OSRH, GPIOx_OSRL) 799
31.6.4 GPIO Three-State Select Registers (GPIOx_TSRH, GPIOx_TSRL) 800
31.6.5 GPIO Open Drain Register (GPIOx_ODR) 800
31.6.6 GPIO Input Register (GPIOx_IR) 801
31.6.7 GPIO Input Select Registers (GPIOx_ISRnH, GPIOx_ISRnL) 801
31.6.8 GPIO Receive Registers (GPIOx_RR1, GPIOx_RR3) 802
31.6.9 Control of GPIO Signals 49–63 802
31.7 GPIO Signal Multiplexing 803

32. Universal Serial Bus Interfaces 809
32.1 USB Overview 809
32.1.1 References 809
32.2 USB 2.0 Configuration 810
32.2.1 USB2PHY0 Control Register (SDR0_USB2PHY0CR) 810
32.2.2 USB2 Host Status Register (SDR0_USB2H0ST) 811
32.2.3 USB2 Host Control Register (SDR0_USB2H0CR) 811
32.2.4 USB2 Device Control Register (SDR0_USB2D0CR) 812
32.3 USB 2.0 PHY Sharing 813
32.3.1 USB2PHY0 Connected to USB2H0 814
32.3.2 USB2PHY0 Connected to USB2D0 814
32.4 USB 2.0 Clocking 815
32.5 USB 2.0 POR Requirements 816
32.6 USB 2.0 Host Interface 816
32.6.1 Overview 816
32.6.2 Functional Description 817
32.6.3 Power Management 817
32.6.3.1 USB EHCI Host Controller 817
32.6.3.2 USB OHCI Host Controller 817
32.6.4 USB 2.0 Host Registers 818
32.6.4.1 Version/Capability Length Register (EHCI0_HCCAPBASE) 820
32.6.4.2 Structural Parameters Register (EHCI0_HCSPARAMS) 820
32.6.4.3 Capability Parameters Register (EHCI0_HCCPARAMS) 822
32.6.4.4 USB Command Register (EHCI0_USBCMD) 823
32.6.4.5 USB Status Register (EHCI0_USBSTS) 825
32.6.4.6 USB Interrupt Enable Register (EHCI0_USBINTR) 827
32.6.4.7 EHCI Frame Index Register (EHCI0_FRINDEX) 828
32.6.4.8 Control Data Structure Segment Register (EHCI0_CRTLSEG) 829
32.6.4.9 Periodic Frame List Base Address Register (EHCI0_PRDLISTB) 829
32.6.4.10 Current Asynchronous List Address Register (EHCI0_ASYNCLSTA) 830
32.6.4.11 Configure Flag Register (EHCI0_CFGFLAG) 830
32.6.4.12 Port Status and Control Register (EHCI0_PORTSC) 831
32.6.4.13 Programmable Microframe Base Value Register (EHCI0_INSNREG0) 835
32.6.4.14 Programmable Packet Buffer Threshold Register (EHCI0_INSNREG1) 836
32.6.4.15 Programmable Packet Buffer Depth Register (EHCI0_INSNREG2) 837
32.6.4.16 Break Memory Transfer Register (EHCI0_INSNREG3) 837
32.6.4.17 Debug Only Register (EHCI0_INSNREG4) 838
32.6.4.18 Revision Register (OHCI0_HCREV) 838
32.6.4.19 Control Register (OHCI0_HCCTRL) 839
32.6.4.20 Command Status Register (OHCI0_HCCMDSTS) 840
32.6.4.21 Interrupt Status Register (OHCI0_HCINTSTS) 842
32.6.4.22 Interrupt Enable Register (OHCI0_HCINTE) 843

32.6.4.23 Interrupt Disable Register (OHCI0_HCINTDIS) 844
32.6.4.24 Host Controller Communications Area Register (OHCI0_HCHCCA) 845
32.6.4.25 Periodic Current Endpoint Descriptor Register (OHCI0_HCPCED) 845
32.6.4.26 Control Head Endpoint Descriptor Register (OHCI0_HCCHED) 846
32.6.4.27 Control Current Endpoint Descriptor Register (OHCI0_HCCTRLCED) 846
32.6.4.28 Bulk Head Endpoint Descriptor Register (OHCI0_HCBULKHED) 847
32.6.4.29 Bulk Current Endpoint Descriptor Register (OHCI0_HCBULKCED) 847
32.6.4.30 Done Head Transfer Descriptor Register (OHCI0_HCDHEAD) 848
32.6.4.31 Frame Interval Register (OHCI0_HCFMINT) 849
32.6.4.32 Frame Remaining Register (OHCI0_HCFMREM) 850
32.6.4.33 Frame Number Register (OHCI0_HCFMNUM) 850
32.6.4.34 Periodic Start Register (OHCI0_HCPRDSTRT) 851
32.6.4.35 Low Speed Threahold Register (OHCI0_HCLSTHRES) 851
32.6.4.36 Root Hub Descriptor A Register (OHCI0_HCRHDESCA) 852
32.6.4.37 Root Hub Descriptor B Register (OHCI0_HCRHDESCB) 854
32.6.4.38 Root Hub Status Register (OHCI0_HCRHSTS) 855
32.6.4.39 Root Hub Port Status Register (OHCI0_HCPRSTS) 856
32.6.4.40 Bridge Control DCR (USB2HMDCR0_BCNTL) 858
32.6.4.41 Bridge Status DCR (USB2HMDCR0_BSTAT) 859
32.6.4.42 Bridge OPB Latency DCR (USB2HMDCR0_LAT) 860
32.6.4.43 Bridge Revision ID DCR (USB2HMDCR0_REVID) 861
32.6.4.44 Bridge Control DCR (USB2HSDCR0_BCNTL) 862
32.6.4.45 Bridge Revision ID DCR (USB2HSDCR0_REVID) 863
32.7 USB 2.0 Device Interface 863
32.7.1 Features 863
32.7.2 Software Interface 865
32.7.3 Memory Map Definitions 865
32.7.4 USB 2.0 Device Registers 866
32.7.4.1 Interrupts for Endpoints 0 and IN 1–3 Register (USB2D0_INTRIN) 867
32.7.4.2 Power Management Register (USB2D0_POWER) 868
32.7.4.3 Function Address Register (USB2D0_FADDR) 869
32.7.4.4 Interrupt Enable for USB2D0_INTRIN Register (USB2D0_INTRINE) 870
32.7.4.5 Interrupts for OUT Endpoints 1 to 3 Register (USB2D0_INTROUT) 870
32.7.4.6 Interrupt enable for USB2D0_INTRUSB Register (USB2D0_INTRUSBE) 871
32.7.4.7 Interrupts for Common USB Register (USB2D0_INTRUSB) 872
32.7.4.8 Interrupt Enable for USB2D0_INTROUT Register (USB2D0_INTROUTE) 873
32.7.4.9 Enable the USB 2.0 Test Modes Register (USB2D0_TSTMODE) 874
32.7.4.10 Endpoint Status/Control Selection Index Register (USB2D0_INDEX) 875
32.7.4.11 Frame Number Register (USB2D0_FRAME) 875
32.7.5 Indexed Registers 876
32.7.5.1 Endpoint 0 Control Status Register (USB2D0_INCSR0) 877
32.7.5.2 Control/Status for IN Endpoint Register (USB2D0_INCSR) 878
32.7.5.3 Maximum Packet Size for IN Endpoint Register (USB2D0_INMAXP) 880
32.7.5.4 Control/Status for OUT Endpoint Register (USB2D0_OUTCSR) 881
32.7.5.5 Maximum Packet Size for OUT Endpoint Register (USB2D0_OUTMAXP) 883
32.7.5.6 Number of Received Bytes in Endpoint 0 FIFO Register (USB2D0_OUTCOUNT0) .. 884
32.7.5.7 Number of Bytes in OUT Endpoint FIFO Register (USB2D0_OUTCOUNT) 884
32.7.6 FIFOs 885

Part V. Reference 887

33. Instruction Set 889

34. Floating Point Instruction Set 891

34.1 Instruction Set Portability 891
34.2 Instruction Formats 891
34.3 Pseudocode 892
34.3.1 Operator Precedence 894
34.4 Register Usage 894
34.5 Floating-Point Instructions 894
34.6 Alphabetical Instruction Listing 894

35. Register Summary 947
35.1 Reserved Fields 947
35.2 Device Control Registers 947
35.3 Memory Mapped Registers 947
35.4 Alphabetical Listing of Chip Control and Peripheral Function Registers 947

36. Signal Summary 971
36.1 Signals Listed Alphabetically 971
36.2 Signals by Interface Category 971

Appendix A. Floating Point Instruction Summary 973
A.1 Instruction Set – Alphabetical 973
A.2 Instructions Sorted by Opcode 977
A.3 Instruction Formats 978
A.3.1 Instruction Fields 979
A.3.2 Instruction Format Diagrams 980
A.3.2.1 I-Form 981
A.3.2.2 B-Form 981
A.3.2.3 SC-Form 981
A.3.2.4 D-Form 981
A.3.2.5 X-Form 982
A.3.2.6 XL-Form 982
A.3.2.7 XFL-Form 983
A.3.2.8 XFX-Form 983
A.3.2.9 X0-Form 983
A.3.2.10 M-Form 983

Index 985

Revision Log 999

Figures

Figure 2-1. Alternate PLB4 Master Priority Register (SDR0_AMP0) 60
Figure 2-2. Alternate PLB3 Master Priority Register 1 (SDR0_AMP1) 62
Figure 2-3. PPC440 CPU Register (SDR0_CP440) 63
Figure 2-4. Master Interrupt Request Register 0 (SDR0_MIRQ0) 64
Figure 2-5. Master Interrupt Request Register 1 (SDR0_MIRQ1) 66
Figure 2-6. Slave Address Pipeline Register (SDR0_SLPIPE0) 67
Figure 2-7. PLB Crossbar Arbiter Interconnection 68
Figure 2-8. PLB4A0 Arbiter 0 Revision ID Register (PLB4A0_REVID) 69
Figure 2-9. PLB4 Arbiter Control Register (PLB4An_ACR) 70
Figure 2-10. PLB4 Error Status Register Low (PLB4An_ESRL) 73
Figure 2-11. PLB4 Error Address Register Low (PLB4An_EARL) 75
Figure 2-12. PLB4 Error Address Register High (PLB4An_EARH) 75
Figure 2-13. PLB4 Crossbar Control Register (PLB4A0_CCR) 76
Figure 2-14. PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL) 77
Figure 2-15. PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH) 77
Figure 2-16. PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0) 78
Figure 2-17. PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1) 80
Figure 2-18. PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG) 82
Figure 2-19. PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR) 83
Figure 2-20. PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR) 84
Figure 2-21. PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID) 84
Figure 2-22. PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL) 85
Figure 2-23. PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH) 85
Figure 2-24. PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0) 86
Figure 2-25. PLB3 to PLB4 Bridge Error Status Register 1 (P3P4BI0_BESR1) 87
Figure 2-26. PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG) 88
Figure 2-27. PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR) 89
Figure 2-28. PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR) 90
Figure 2-29. PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID) 90
Figure 2-30. PLB4 to OPB Bridge Error Status Register 0 (PLB42OPBx_BESR0) 91
Figure 2-31. PLB4 to OPB Bridge Error Address Register Low (PLB42OPBx_BEARL) 93
Figure 2-32. PLB4 to OPB Bridge Error Address Register High (PLB42OPBx_BEARH) 93
Figure 2-33. PLB4 to OPB Bridge Error Status Register 1 (PLB42OPBx_BESR1) 94
Figure 2-34. PLB4 to OPB Bridge Configuration Register (PLB42OPBx_CFG) 95
Figure 2-35. PLB4 to OPB Bridge Burst Latency Timer Register (PLB42OPBx_LATENCY) 96
Figure 2-36. PLB4 to OPB Bridge Revision ID Register (PLB42OPBx_REVID) 96
Figure 2-37. PLB3 Arbiter 0 Revision ID Register (PLB3A0_REVID) 97
Figure 2-38. PLB3 Error Status Register (PLB3A0_BESR) 97
Figure 2-39. PLB3 Error Address Register (PLB3A0_BEAR) 98
Figure 2-40. PLB3 Arbiter Control Register (PLB3A0_ACR) 98

Figure 2-41. PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0) 100
Figure 2-42. PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR) 101
Figure 2-43. PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID) 101
Figure 2-44. PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1) 102
Figure 2-45. OPB Arbiter Priority Register (OPBAx_PR) 104
Figure 2-46. OPB Arbiter Control Register (OPBAx_CR) 104
Figure 2-47. OPB-to-PLB4 Bridge Control Register (OPB2PLB40_BCTRL) 105
Figure 2-48. OPB-to-PLB4 Bridge Status Register (OPB2PLB40_BSTAT) 106
Figure 2-49. OPB-to-PLB4 Bridge Error Address Register Low (OPB2PLB40_BEARL) 106
Figure 2-50. OPB-to-PLB4 Bridge Error Address Register High (OPB2PLB40_BEARH) 106
Figure 2-51. OPB-to-PLB4 Bridge Revision ID Register (OPB2PLB40_REVID) 107
Figure 4-1. Floating-Point Registers (FPR0:FPR31) 142
Figure 4-2. Floating-Point Status and Control Register (FPSCR) 143
Figure 4-3. Approximation to Real Numbers 147
Figure 4-4. Selection of z1 and z2 151
Figure 7-1. PPC440EPx/GRx Power-on Reset Process 170
Figure 7-2. PPC440EPx/GRx Reset 171
Figure 7-3. Initial Configuration Register (CPR0_ICFG) 176
Figure 7-4. SDR Configuration Address Register (SDR0_CFGADDR) 176
Figure 7-5. SDR Configuration Data Register (SDR0_CFGDATA) 176
Figure 7-6. Electronic Chip ID Register 0 (SDR0_ECID0) 177
Figure 7-7. Electronic Chip ID Register 1 (SDR0_ECID1) 177
Figure 7-8. Electronic Chip ID Register 1 (SDR0_ECID2) 177
Figure 7-9. Electronic Chip ID Register 3 (SDR0_ECID3) 177
Figure 7-10. Pin Function Control Register 0 (SDR0_PFC0) 177
Figure 7-11. Pin Function Control Register 1 (SDR0_PFC1) 178
Figure 7-12. GPIO Multiplexing Register (SDR0_PFC4) 179
Figure 7-13. Slave Address Pipeline Register (SDR0_SLPIPE0) 179
Figure 7-14. Soft Reset Register 0 (SDR0_SRST0) 179
Figure 7-15. Soft Reset Register 1(SDR0_SRST1) 180
Figure 7-16. Miscellaneous Function Register (SDR0_MFR) 181
Figure 7-17. DDR Configuration Register (SDR0_DDRCFG) 182
Figure 8-1. IIC Bootstrap Controller Flow 194
Figure 8-2. Pin Strapping Register (SDR0_PINSTP) 197
Figure 8-3. Serial Device Controller Settings Register (SDR0_SDCS0) 198
Figure 8-4. Serial Device Strap Register 0 (SDR0_SDSTP0) 198
Figure 8-5. Serial Device Strap Register 1 (SDR0_SDSTP1) 200
Figure 8-6. Serial Device Strap Register 2 (SDR0_SDSTP2) 201
Figure 8-7. Serial Device Strap Register 3 (SDR0_SDSTP3) 202
Figure 8-8. Serial Device Strap Register 0 (SDR0_CUST0) 203
Figure 8-9. Custom Configuration Register 1 (SDR0_CUST1) 204

Figure 8-10. EBC Configuration Register (SDR0_EBC0) 204
Figure 9-1. Cascaded UIC Organization 205
Figure 9-2. UIC0 Status Register (UIC0_SR) 210
Figure 9-3. UIC1 Status Register (UIC1_SR) 212
Figure 9-4. UIC2 Status Register (UIC2_SR) 215
Figure 9-5. UIC0 Enable Register (UIC0_ER) 216
Figure 9-6. UIC1 Enable Register (UIC1_ER) 218
Figure 9-7. UIC2 Enable Register (UIC2_ER) 220
Figure 9-8. UIC0 Critical Register (UIC0_CR) 221
Figure 9-9. UIC1 Critical Register (UIC1_CR) 223
Figure 9-10. UIC2 Critical Register (UIC2_CR) 226
Figure 9-11. UIC0 Polarity Register (UIC0_PR) 227
Figure 9-12. UIC1 Polarity Register (UIC1_PR) 229
Figure 9-13. UIC2 Polarity Register (UIC2_PR) 231
Figure 9-14. UIC0 Trigger Register (UIC0_TR) 232
Figure 9-15. UIC1 Trigger Register (UIC1_TR) 235
Figure 9-16. UIC2 Trigger Register (UIC2_TR) 237
Figure 9-17. UIC0 Masked Status Register (UIC0_MSR) 238
Figure 9-18. UIC1 Masked Status Register (UIC1_MSR) 240
Figure 9-19. UIC2 Maked Status Register (UIC2_MSR) 242
Figure 9-20. UIC0 Vector Configuration Register (UIC0_VCR) 243
Figure 9-21. UIC1 Vector Configuration Register (UIC1_VCR) 244
Figure 9-22. UIC Vector Register (UIC0_VR) 244
Figure 9-23. UIC1 Vector Register (UIC1_VR) 245
Figure 11-1. Condition Register (CR) 260
Figure 13-1. Time Base Counter and Compare Register 266
Figure 13-2. GPT Time Base Counter Register (GPT0_TBC) 267
Figure 13-3. GPT Interrupt Mask Register (GPT0_IM) 268
Figure 13-4. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC) 269
Figure 13-5. GPT Interrupt Enable Register (GPT0_IE) 269
Figure 13-6. GPT Compare Timer Register (GPT0_COMP0 - GPT0_COMP6) 270
Figure 13-7. GPT Compare Mask Register (GPT0_MASK0:GPT0_MASK6) 270
Figure 13-8. Down Count Timer Register (GPT0_DCT0) 270
Figure 13-9. Down Count Timer Register (GPT0_DCIS) 271
Figure 14-1. PPC440EPx/GRx System Clocking 274
Figure 14-2. Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR) 284
Figure 14-3. Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA) 284
Figure 14-4. Clocking Update Register (CPR0_CLKUPD) 285
Figure 14-5. PLL Control Register (CPR0_PLLC0) 285
Figure 14-6. PLL Divisor Register (CPR0_PLLD) 286
Figure 14-7. Primary A Divisor Register (CPR0_PRIMAD0) 287

Figure 14-8. Primary B Divisor Register (CPR0_PRIMBD0) 288
Figure 14-9. OPB Clock Divisor Register (CPR0_OPBD0) 288
Figure 14-10. Peripheral Clock Divisor Register (CPR0_PERD0) 289
Figure 14-11. MAL Clock Divisor Register (CPR0_MALD) 289
Figure 14-12. Sync PCI Clock Divisor Register (CPR0_SPCID) 289
Figure 15-1. CPM0 Enable Register (CPM0_ER) 292
Figure 15-2. CPM1 Enable Register (CPM1_ER) 293
Figure 15-3. CPM0 Force Register (CPM0_FR) 293
Figure 15-4. CPM1 Force Register (CPM1_FR) 294
Figure 15-5. CPM Status Register (CPM0_SR) 295
Figure 15-6. CPM1 Status Register (CPM1_SR) 296
Figure 17-1. Security Function Block Diagram 300
Figure 17-2. CRYP0 SDR Control Register (SDR0_CRYP0) 312
Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST) 313
Figure 17-4. PE Source Address Register (CRYP0_PE_SOURCE) 319
Figure 17-5. PE Destination Address Register (CRYP0_PE_DEST) 319
Figure 17-6. PE SA Address Register (CRYP0_PE_SA) 320
Figure 17-7. PE Length Register (CRYP0_PE_LENGTH) 321
Figure 17-8. PE DMA Configuration Register (CRYP0_PE_DMA_CF) 322
Figure 17-9. PE DMA Status Register (CRYP0_PE_DMA_ST) 324
Figure 17-10. PE Packet Descriptor Ring Base Address Register (CRYP0_PE_PDR_BA) 326
Figure 17-11. PE Packet Descriptor Ring Base Address Register (CRYP0_PE_PDR_BA) 326
Figure 17-12. PE Packet Result Ring Base Address Register (CRYP0_PE_RDR_BA) 327
Figure 17-13. PE Ring Size and Offset Register (CRYP0_PE_RING_S) 328
Figure 17-14. PE Ring Poll Register (CRYP0_PE_RING_P) 329
Figure 17-15. PE Internal Ring Status Register (CRYP0_PE_I_RING) 330
Figure 17-16. PE External Ring Status Register (CRYP0_PE_I_RING) 331
Figure 17-17. PE I/O Threshold Register (CRYP0_PE_IO_THR) 331
Figure 17-18. PE Gather Particle Ring Base Address Register (CRYP0_PE_GATH) 332
Figure 17-19. PE Scatter Particle Ring Base Address Register (CRYP0_PE_SCAT) 332
Figure 17-20. PE Particle Ring Size Register (CRYP0_PE_PT_S) 333
Figure 17-21. PE Particle Ring Configuration Register (CRYP0_PE_PT_CFG) 333
Figure 17-22. PE Particle Descriptor Source Address Register (CRYP0_PE_PR_SCA) 334
Figure 17-23. PE Particle Descriptor Source Control Register (CRYP0_PE_PR_SCC) 335
Figure 17-24. PE Particle Destination Address Register (CRYP0_PE_PR_DTA) 336
Figure 17-25. PE Particle Destination Control Register (CRYP0_PE_PR_DTC) 337
Figure 17-26. SA Command 0 Register (CRYP0_SA_CMD_0) 338
Figure 17-27. SA Command 1 Register (CRYP0_SA_CMD_1) 342
Figure 17-28. SA Key x Low/High Registers (CRYP0_SA_KEYx_L/H) 346
Figure 17-29. SA Inner Hash Digest x Registers (CRYP0_SA_IH_Dx) 347
Figure 17-30. SA Inner Hash Digest x Registers (CRYP0_SA_OH_Dx) 348

Figure 17-31. SA IPsec SPI Register (CRYP0_SA_SPI) 349
Figure 17-32. SA IPsec Sequence Number Register (CRYP0_SA_SEQ) 350
Figure 17-33. SA IPsec Sequence Number Mask High and Low Registers (CRYP0_SA_SEQMKH/L 351
Figure 17-34. SA Nonce Value Register (CRYP0_SA_NONCE) 351
Figure 17-35. SA Pointer Register (CRYP0_SA_PNTR) 352
Figure 17-36. SA ARC4 i and J Pointer Register (CRYP0_SA_ARC4IJ) 352
Figure 17-37. SA ARC4 State Address Pointer Register (CRYP0_SA_ARC4SB) 353
Figure 17-38. SA Initialization Vector Registers (CRYP0_SA_IV_x) 353
Figure 17-39. SA Hash Byte Count Register (CRYP0_SA_HASH_B) 354
Figure 17-40. SA Inner Hash x (mirror of CRYP0_SA_IH_Dx) Registers (CRYP0_SA_IH_x) 354
Figure 17-41. SA ICV x—HMAC Result Registers (CRYP0_SA_ICV_x) 355
Figure 17-42. TRNG Output Register (CRYP0_TRNG_DATA) 356
Figure 17-43. TRNG Status Register (CRYP0_TRNG_STAT) 356
Figure 17-44. TRNG Control Register (CRYP0_TRNG_CTRL) 357
Figure 17-45. TRNG Entropy A Register (CRYP0_TRNG_ENTA) 358
Figure 17-46. TRNG Entropy B Register (CRYP0_TRNG_ENTB) 359
Figure 17-47. TRNG Test Seed x Registers (CRYP0_TRNG_Xx) 359
Figure 17-48. TRNG Counter Register (CRYP0_TRNG_CNTR) 360
Figure 17-49. TRNG Alarm Counter Register (CRYP0_TRNG_ALRM) 360
Figure 17-50. TRNG Configuration Register (CRYP0_TRNG_CFG) 361
Figure 17-51. TRNG Test Read of LFSR 0 Low Register (CRYP0_TRNG_LF0L) 362
Figure 17-52. TRNG Test Read of LFSR 0 High Register (CRYP0_TRNG_LF0H) 362
Figure 17-53. TRNG Test Read of LFSR 1 Low Register (CRYP0_TRNG_LF1L) 363
Figure 17-54. TRNG Test Read of LFSR 1 High Register (CRYP0_TRNG_LF1H) 363
Figure 17-55. TRNG Triple DES Key 0 Low/High Registers (CRYP0_TRNG_K0_L/H) 363
Figure 17-56. TRNG Triple DES Key 1 Low/High Registers (CRYP0_TRNG_K1_L/H) 364
Figure 17-57. TRNG Initialization Vector Low/High Registers (CRYP0_TRNG_IV_L/H) 364
Figure 17-58. PKA x Vector Address Register (CRYP0_PKA_x_PTR) 365
Figure 17-59. PKA x Vector Length Register (CRYP0_PKA_x_LEN) 365
Figure 17-60. PKA Shift Register (CRYP0_PKA_SHIFT) 366
Figure 17-61. PKA Function Code Register (CRYP0_PKA_FUNC) 367
Figure 17-62. Mathematical Operations 368
Figure 17-63. Operations Description 368
Figure 17-64. PKA Comparison Result Register (CRYP0_PKA_COMP) 370
Figure 17-65. PKA Quotient MSW Register (CRYP0_PKA_DIV) 371
Figure 17-66. PKA Remainder MSW Register (CRYP0_PKA_MOD) 372
Figure 17-67. Interrupt Unmasked and Masked Status Registers (CRYP0_INT_UNMSK/MSK) 374
Figure 17-68. Interrupt Mask Register (CRYP0_INT_EN) 375
Figure 17-69. Interrupt Configuration Register (CRYP0_INT_CFG) 376
Figure 17-70. Interrupt Force Descriptor Read Register (CRYP0_INT_DESRD) 377
Figure 17-71. Interrupt Descriptor Count Register (CRYP0_INT_DESCT) 377

Figure 17-72. Device Control Register (CRYP0_DC_CTRL) 378
Figure 17-73. Device ID Register (CRYP0_DC_DEVID) 378
Figure 17-74. Device Information Register (CRYP0_DC_DEVINF) 379
Figure 17-75. DMA Source Address Register (CRYP0_DMA_USRC) 380
Figure 17-76. DMA Destintion Address Register (CRYP0_DMA_UDST) 380
Figure 17-77. DMA Command Register (CRYP0_DMA_UCMD) 381
Figure 17-78. DMA Configuration/Status Register (CRYP0_DMA_CFG) 382
Figure 17-79. PRNG Status Register (CRYP0_PRNG_STAT) 383
Figure 17-80. PRNG Control Register (CRYP0_PRNG_CTRL) 384
Figure 17-81. PRNG Seed Value L/H Registers (CRYP0_PRNG_SDL/H) 385
Figure 17-82. PRNG Key x Low/High Registers (CRYP0_PRNG_KEYx_L/H) 385
Figure 17-83. PRNG Result x Registers (CRYP0_PRNG_RSx) 386
Figure 17-84. PRNG LFSR Low/High Registers (CRYP0_PRNG_LFL/H) 386
Figure 18-1. KASUMI Block Diagram 390
Figure 18-2. Data Input/Output0 Register (KASU0_DATAIN0, KASU0_DATAOUT0) 394
Figure 18-3. Data Input/Output0 Register (KASU0_DATAIN1, KASU0_DATAOUT1) 394
Figure 18-4. Control/Status Register (KASU0_CTRL, KASU0_STAT) 395
Figure 18-5. Mode Register (KASU0_MODE) 396
Figure 18-6. Key Register (KASU0_KEY0) 397
Figure 18-7. Key Register (KASU0_KEY1) 397
Figure 18-8. Key Register (KASU0_KEY2) 397
Figure 18-9. Key Register (KASU0_KEY3) 397
Figure 18-10. Count Register (KASU0_COUNT) 398
Figure 18-11. Configuration Register (KASU0_CONFIG) 398
Figure 18-12. Fresh Register (KASU0_FRESH) 399
Figure 19-1. DDR SDRAM Configuration Address Register (DDR0_CFGADDR) 410
Figure 19-2. DDR SDRAM Configuration Data Register (DDR0_CFGDATA) 410
Figure 19-3. DDR0_00 Register 412
Figure 19-4. DDR0_01 Register 413
Figure 19-5. DDR0_02 Register 414
Figure 19-6. DDR0_03 Register 415
Figure 19-7. DDR0_04 Register 416
Figure 19-8. DDR0_05 Register 416
Figure 19-9. DDR0_06 Register 417
Figure 19-10. DDR0_07 Register 418
Figure 19-11. DDR0_08 Register 419
Figure 19-12. DDR0_09 Register 420
Figure 19-13. DDR0_10 Register 421
Figure 19-14. DDR0_11 Register 422
Figure 19-15. DDR0_12 Register 422
Figure 19-16. DDR0_14 Register 423

Figure 19-17. DDR0_17 Register 424
Figure 19-18. DDR0_18 Register 424
Figure 19-19. DDR0_19 Register 425
Figure 19-20. DDR0_20 Register 425
Figure 19-21. DDR0_21 Register 426
Figure 19-22. DDR0_22 Register 427
Figure 19-23. DDR0_23 Register 428
Figure 19-24. DDR0_24 Register 429
Figure 19-25. DDR0_25 Register 429
Figure 19-26. DDR0_26 Register 430
Figure 19-27. DDR0_27 Register 430
Figure 19-28. DDR0_28 Register 430
Figure 19-29. DDR0_31 Register 430
Figure 19-30. DDR0_32 Register 431
Figure 19-31. DDR0_33 Register 431
Figure 19-32. DDR0_34 Register 431
Figure 19-33. DDR0_35 Register 431
Figure 19-34. DDR0_36 Register 431
Figure 19-35. DDR0_37 Register 431
Figure 19-36. DDR0_38 Register 432
Figure 19-37. DDR0_39 Register 432
Figure 19-38. DDR0_40 Register 432
Figure 19-39. DDR0_41 Register 432
Figure 19-40. DDR0_42 Register 433
Figure 19-41. DDR0_43 Register 433
Figure 19-42. DDR0_44 Register 434
Figure 19-43. DQS Gating 434
Figure 19-44. DQS Read Timing 440
Figure 19-45. Data Capture Logic 442
Figure 19-46. Read Data Timing (Earliest) 443
Figure 19-47. Read Data Timing (Latest) 443
Figure 19-48. DRAM DQS Arrival Time Requirements 444
Figure 19-49. DQS Write Timing 445
Figure 19-50. Write Data and DQS Relationship 446
Figure 20-1. Memory Configuration (SRAM0_SB0CR) 448
Figure 20-2. Bus Error Address Register (SRAM0_BEAR) 448
Figure 20-3. Bus Error Status Register 0 (SRAM0_BESR0) 449
Figure 20-4. Bus Error Status Register 1 (SRAM0_BESR1) 451
Figure 20-5. Power Management Register (SRAM0_PMEG) 452
Figure 20-6. SRAM Internal Core Device ID Register (SRAM0_CID) 452
Figure 20-7. SRAM Revision ID Register (SRAM0_REVID) 453

Figure 20-8. Data Parity Checking Register (SRAM0_DPC) 453
Figure 21-1. PCI Bridge Block Diagram 456
Figure 21-2. PLB-to-PCI Half-Bridge Block Diagram 457
Figure 21-3. PCI-to-PLB Half-Bridge Block Diagram 458
Figure 21-4. Arbitration Structure 458
Figure 21-5. PMM Register Sets Map PLB Address Space to PCI Address Space 461
Figure 21-6. PTM Register Sets Map PCI Address Space to PLB Address Space 462
Figure 21-7. PCI Control Register (SDR0_PCI0) 470
Figure 21-8. PMM 0 Local Address Register (PCIL0_PMM0LA) 473
Figure 21-9. PMM 0 Mask/Attribute Register (PCIL0_PMM0MA) 473
Figure 21-10. PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA) 474
Figure 21-11. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA) 474
Figure 21-12. PMM 1 Local Address Register (PCIL0_PMM1LA) 474
Figure 21-13. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA) 474
Figure 21-14. PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA) 475
Figure 21-15. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA) 475
Figure 21-16. PMM 2 Local Address Register (PCIL0_PMM2LA) 475
Figure 21-17. PMM 2 Mask/Attribute Register (PCIL0_PMM2MA) 475
Figure 21-18. PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA) 476
Figure 21-19. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA) 476
Figure 21-20. PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS) 476
Figure 21-21. PTM 2 Local Address Register (PCIL0_PTM1LA) 477
Figure 21-22. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS) 477
Figure 21-23. PTM 2 Local Address Register (PCIL0_PTM2LA) 477
Figure 21-24. PCI Configuration Address Register (PCIC0_CFGADDR) 478
Figure 21-25. PCI Configuration Data Register (PCIC0_CFGDATA) 478
Figure 21-26. PCI Vendor ID Register (PCIC0_VENDID) 479
Figure 21-27. PCI Device ID Register (PCIC0_DEVID) 479
Figure 21-28. PCI Command Register (PCIC0_CMD) 479
Figure 21-29. PCI Status Register (PCIC0_STATUS) 480
Figure 21-30. PCI Revision ID Register (PCIC0_REVID) 481
Figure 21-31. PCI Class Register (PCIC0_CLS) 481
Figure 21-32. PCI Cache Line Size Register (PCIC0_CACHELS) 482
Figure 21-33. PCI Latency Timer Register (PCIC0_LATTIM) 482
Figure 21-34. PCI Header Type Register (PCIC0_HDTYPE) 482
Figure 21-35. PCI Built-in Self Test Control Register (PCIC0_BIST) 483
Figure 21-36. PCI PTM 1 BAR Register (PCIC0_PTM1BAR) 483
Figure 21-37. PCI PTM 2 BAR Register (PCIC0_PTM2BAR) 483
Figure 21-38. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID) 484
Figure 21-39. PCI Subsystem ID Register (PCIC0_SBSYSID) 484
Figure 21-40. PCI Capabilities Pointer (PCIC0_CAP) 484

Figure 21-41. PCI Interrupt Line Register (PCIC0_INTLN) 484
Figure 21-42. PCI Interrupt Pin Register (PCIC0_INTPN) 485
Figure 21-43. PCI Minimum Grant Register (PCIC0_MINGNT) 485
Figure 21-44. PCI Maximum Latency Register (PCIC0_MAXLTNCY) 485
Figure 21-45. PCI Interrupt Control/Status Register 485
Figure 21-46. PCI Error Enable Register (PCIC0_ERREN) 485
Figure 21-47. PCI Error Status Register (PCIC0_ERRSTS) 486
Figure 21-48. PCI Bridge Options 1 Register (PCIC0_BRDGOPT1) 487
Figure 21-49. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0) 488
Figure 21-50. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1) 489
Figure 21-51. PLB Slave Error Address Register (PCIC0_PLBBEAR) 490
Figure 21-52. PCI Capability Identifier (PCIC0_CAPID) 490
Figure 21-53. PCI Next Item Pointer (PCIC0_NEXTIPTR) 490
Figure 21-54. Power Management Capabilities Register (PCIC0_PMC) 491
Figure 21-55. Power Management Control/Status Register (PCIC0_PMCSR) 491
Figure 21-56. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBSE) 492
Figure 21-57. PCI Data Register (PCIC0_DATA) 492
Figure 21-58. PCI Bridge Options 2 Register (PCIC0_BRDGOPT2) 492
Figure 21-59. Power Management State Change Request Register (PCIC0_PMSCRR) 493
Figure 21-60. Example Address Map 499
Figure 21-61. PCI Master Burst Read From SDRAM 503
Figure 21-62. PCI Master Burst Read From SDRAM (Continued) 504
Figure 21-63. PCI Master Burst Read From SDRAM (Continued) 505
Figure 21-64. PCI Master Burst Read From SDRAM (Continued) 506
Figure 21-65. PCI Master Burst Write To SDRAM 507
Figure 21-66. PCI Master Burst Write To SDRAM (Continued) 508
Figure 21-67. PCI Master Burst Write To SDRAM (Continued) 509
Figure 21-68. PCI Master Burst Write To SDRAM (Continued) 510
Figure 21-69. CPU Read From PCI Memory Slave, Nonprefetching 511
Figure 21-70. CPU Read From PCI Memory Slave, Nonprefetching (Continued) 512
Figure 21-71. CPU Read From PCI Memory Slave, Prefetching 513
Figure 21-72. CPU Read From PCI Memory Slave, Prefetching (Continued) 514
Figure 21-73. CPU Read From PCI Memory Slave, Prefetching (Continued) 515
Figure 21-74. CPU Read From PCI Memory Slave, Prefetching (Continued) 516
Figure 21-75. CPU Write To PCI Memory Slave 517
Figure 21-76. CPU Write To PCI Memory Slave (Continued) 518
Figure 21-77. CPU Write To PCI Memory Slave (Continued) 519
Figure 21-78. PCI Memory To SDRAM DMA Transfer 520
Figure 21-79. PCI Memory To SDRAM DMA Transfer (Continued) 521
Figure 21-80. SDRAM To PCI Memory DMA Transfer 522
Figure 21-81. SDRAM To PCI Memory DMA Transfer (Continued) 523

Figure 21-82. SDRAM To PCI Memory DMA Transfer (Continued) 524
Figure 21-83. PCI Master Burst Read From SDRAM 525
Figure 21-84. PCI Master Burst Read From SDRAM (Continued) 526
Figure 21-85. PCI Master Burst Read From SDRAM (Continued) 527
Figure 21-86. PCI Master Burst Read From SDRAM (Continued) 528
Figure 21-87. PCI Master Burst Read From SDRAM (Continued) 529
Figure 21-88. PCI Master Burst Read From SDRAM (Continued) 530
Figure 21-89. PCI Master Burst Read From SDRAM (Continued) 531
Figure 21-90. PCI Master Burst Write To SDRAM 532
Figure 21-91. PCI Master Burst Write To SDRAM (Continued) 533
Figure 21-92. PCI Master Burst Write To SDRAM (Continued) 534
Figure 21-93. PCI Master Burst Write To SDRAM (Continued) 535
Figure 21-94. PCI Master Burst Write To SDRAM (Continued) 536
Figure 21-95. PCI Master Burst Write To SDRAM (Continued) 537
Figure 21-96. CPU Read From PCI Memory Slave, Nonprefetching 538
Figure 21-97. CPU Read From PCI Memory Slave, Nonprefetching (Continued) 539
Figure 21-98. CPU Read From PCI Memory Slave, Prefetching 540
Figure 21-99. CPU Read From PCI Memory Slave, Prefetching (Continued) 541
Figure 21-100. CPU Read From PCI Memory Slave, Prefetching (Continued) 542
Figure 21-101. CPU Read From PCI Memory Slave, Prefetching (Continued) 543
Figure 21-102. CPU Write To PCI Memory Slave 544
Figure 21-103. CPU Write To PCI Memory Slave (Continued) 545
Figure 21-104. CPU Write To PCI Memory Slave (Continued) 546
Figure 21-105. CPU Write To PCI Memory Slave (Continued) 547
Figure 21-106. PCI Memory To SDRAM DMA Transfer 548
Figure 21-107. PCI Memory To SDRAM DMA Transfer (Continued) 549
Figure 21-108. PCI Memory To SDRAM DMA Transfer (Continued) 550
Figure 21-109. SDRAM To PCI Memory DMA Transfer 551
Figure 21-110. SDRAM To PCI Memory DMA Transfer (Continued) 552
Figure 21-111. SDRAM To PCI Memory DMA Transfer (Continued) 553
Figure 22-1. EBC Signals 555
Figure 22-2. Attachment of Devices of Various Widths to the Peripheral Data Bus 557
Figure 22-3. Single Read Transfer 559
Figure 22-4. Single Write Transfer 560
Figure 22-5. Burst Read Transfer 562
Figure 22-6. Burst Write Transfer 563
Figure 22-7. Device-Paced Single Read Transfer 564
Figure 22-8. Device-Paced Single Write Transfer 565
Figure 22-9. Device-Paced Burst Read Transfer 566
Figure 22-10. Device-Paced Burst Write Transfer 568
Figure 22-11. Sample External Bus Master System 569

Figure 22-12. External Master Arbitration, Single Read and Single Write 572
Figure 22-13. External Master Burst Read 573
Figure 22-14. External Master Burst Write 574
Figure 22-15. EBC Configuration Address Register (EBC0_CFGADDR) 574
Figure 22-16. EBC Configuration Data Register (EBC0_CFGDATA) 575
Figure 22-17. EBC Configuration Register (EBC0_CFG) 576
Figure 22-18. Peripheral Bank Configuration Registers (EBC0_B0CR–EBC0_B5CR) 577
Figure 22-19. Peripheral Bank Access Parameters (EBC0_B0AP–EBC0_B5AP) 578
Figure 22-20. Peripheral Bus Error Address Register (EBC0_BEAR) 580
Figure 22-21. Peripheral Bus Error Status Register 0 (EBC0_BESR0) 581
Figure 23-1. Typical System Application 584
Figure 23-2. IPL/SPL Boot Sequence using Boot-from-NAND 587
Figure 23-3. Column Parity Generation 594
Figure 23-4. Line/Row Parity Generation 595
Figure 23-5. Basic External Device Timing Controls 596
Figure 23-6. Command write cycle 597
Figure 23-7. Address Cycle 597
Figure 23-8. Single (raw) Read Cycle 598
Figure 23-9. Burst (Page) Read Cycle 598
Figure 23-10. Single (raw) Write Cycle 599
Figure 23-11. Burst (Page) Write Cycle 599
Figure 23-12. NAND Flash Address Register (NDFC0_ADDR) 601
Figure 23-13. NAND Flash Command Register (NDFC0_CMD) 601
Figure 23-14. NAND Flash Data Register (NDFC0_DATA) 602
Figure 23-15. NAND Flash ECC Calculation Register (NDFC0_ECC0:NDFC0_ECC7) 603
Figure 23-16. NAND Flash Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR) 604
Figure 23-17. NAND Flash Configuration Register (NDFC0_CR) 604
Figure 23-18. NAND Flash Status Register (NDFC0_SR) 606
Figure 23-19. NAND Flash Direct Hardware Control Register (NDFC0_HWCTL) 607
Figure 23-20. NAND Flash Revision ID Register (NDFC0_REVID) 608
Figure 24-1. DMA to PLB4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3) 613
Figure 24-2. DMA to PLB4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3) 615
Figure 24-3. DMA to PLB4 Source Address High Registers (DMA2P40-SAH0-DMA2P40-SAH3) 616
Figure 24-4. DMA to PLB4 Source Address Low Registers (DMA2P40-SAL0-DMA2P40-SAL3) 616
Figure 24-5. DMA to PLB4 Destination Address High Registers (DMA2P40_DAH0-DMA2P40_DAH3) 616
Figure 24-6. DMA to PLB4 Destination Address Low Registers (DMA2P40_DAL0-DMA2P40_DAL3) ..616
Figure 24-7. DMA to PLB4 Scatter/Gather Desc Adr High Reg (DMA2P40_SGH0-DMA2P40_SGH3) .617
Figure 24-8. DMA to PLB4 Scatter/Gather Desc Adr Low Reg (DMA2P40_SGL0-DMA2P40_SGL3) ...617
Figure 24-9. DMA to PLB4 Status Register (DMA2P40_SR) 617
Figure 24-10. DMA to PLB4 Scatter/Gather Command Register (DMA2P40_SGC) 618
Figure 24-11. DMA to PLB4 Sleep Mode Register (DMA2P40_SLP) 619

Figure 24-12. Scatter/Gather Descriptor Table 622
Figure 24-13. DMA to PLB 3 External Bus Controller Signals 627
Figure 24-14. DMA to PLB3 Polarity Configuration Register (DMA2P30_POL) 630
Figure 24-15. DMA to PLB3 Sleep Mode Register (DMA2P30_SLP) 631
Figure 24-16. DMA to PLB3 Status Register (DMA2P30_SR) 632
Figure 24-17. DMA to PLB3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3) 632
Figure 24-18. DMA to PLB3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3) 634
Figure 24-19. DMA to PLB3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3) 634
Figure 24-20. DMA to PLB3 Count Registers (DMA2P30_CT0–DMA2P30_CT3) 635
Figure 24-21. DMA to PLB3 Scatter/Gather Desc. Addr. Regs (DMA2P30_SG0–DMA2P30_SG3) 635
Figure 24-22. DMA to PLB3 Scatter/Gather Command Register (DMA2P30_SGC) 636
Figure 24-23. DMA to PLB3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3) 636
Figure 24-24. DMA to PLB3 Address Decode Register (DMA2P30_ADR) 636
Figure 24-25. Peripheral to Memory DMA Transfers 641
Figure 24-26. Memory to Peripheral DMA Transfers 642
Figure 25-1. General PPC440EPx/GRx Structure 646
Figure 25-2. MAL Internal Structure 647
Figure 25-3. Transmit Operations 649
Figure 25-4. Receive Operations 650
Figure 25-5. MAL Receive Burst Length Register (SDR0_MALRBL) 651
Figure 25-6. MAL Receive Bus Size Register (SDR0_MALRBS) 651
Figure 25-7. MAL Transmit Burst Length Register (SDR0_MALTBL) 652
Figure 25-8. MAL Transmit Bus Size Register (SDR0_MALTBS) 652
Figure 25-9. Buffer Descriptor Structure 653
Figure 25-10. Packet Memory Structure 654
Figure 25-11. Error Status Register Field 665
Figure 25-12. MAL Error Processing 667
Figure 25-13. MAL Configuration Register (MAL0_CFG) 669
Figure 25-14. TX Channel_Active Set Register (MAL0_TXCASR) 670
Figure 25-15. TX Channel Active Reset Register (MAL0_TXCARR) 670
Figure 25-16. RX Channel_Active Set Register (MAL0_RXCASR) 671
Figure 25-17. RX Channel_Active Reset Register (MAL0_RXCARR) 671
Figure 25-18. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR) 671
Figure 25-19. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR) 672
Figure 25-20. MAL Error Status Register (MAL0_ESR) 672
Figure 25-21. MAL Interrupt Enable Register (MAL0_IER) 674
Figure 25-22. TX Descriptor Error Interrupt Register (MAL0_TXDEIR) 675
Figure 25-23. RX Descriptor Error Interrupt Register (MAL0_RXDEIR) 675
Figure 25-24. TX PLB Attribute Register (MAL0_TXTATTRR) 675
Figure 25-25. RX PLB Attribute Register (MAL0_RXTATTRR) 675
Figure 25-26. TX Descriptor Base Address Register (MAL0_TXBADDR) 676

Figure 25-27. RX Descriptor Base Address Register (MAL0_RXBADDR) 676
Figure 25-28. TX Channel Table Pointer Register (MAL0_TXCTPxR) 677
Figure 25-29. RX Channel Table Pointer Register (MAL0_RXCTPxR) 677
Figure 25-30. RX Channel Buffer Size Register (MAL0_RCBSx) 677
Figure 25-31. Interrupt Coalescing Control Transmit Register (SDR0_ICCRTX) 678
Figure 25-32. Interrupt Coalescing Control Receive Register (SDR0_ICCRRX) 679
Figure 25-33. Interrupt Coalescing Timer Register Transmit Channel x (SDR0_ICTRTXx) 680
Figure 25-34. Interrupt Coalescing Timer Register Receive Channel x (SDR0_ICTRRXx) 680
Figure 25-35. Interrupt Coalescing Status Register Transmit Channel x (SDR0_ICSRTXx) 680
Figure 25-36. Interrupt Coalescing Status Register Receive Channel x (SDR0_ICSRRXx) 680
Figure 26-1. ZMII andRGMII Bridge Configurations 682
Figure 26-2. EMAC to PHY Using MII 683
Figure 26-3. EMAC to PHY Using SMII 684
Figure 26-4. Function Enable Register (ZMII0_FER) 685
Figure 26-5. Speed Selection Register (ZMII0_SSR) 685
Figure 26-6. SMII Status Register (ZMII0_SMIISR) 686
Figure 26-7. GMII 24 Pins + 2 688
Figure 26-8. EMAC to PHY Using RGMII Bridge 689
Figure 26-9. RGMII Function Enable Register (RGMII0_FER) 690
Figure 26-10. Speed Selection Register (RGMII0_SSR) 690
Figure 26-11. Ethernet PLL Configuration Register (SDR0_PFC2) 691
Figure 27-1. EMAC in a Typical Ethernet Application 694
Figure 27-2. Internal EMAC Structure 696
Figure 27-3. EMAC Loopback Modes 698
Figure 27-4. EMACx RX Status Register (SDR0_EMACxRXST) 699
Figure 27-5. EMACx TX Status Register (SDR0_EMACxTXST) 701
Figure 27-6. EMACx Reject Count Register (SDR0_EMACxREJCNT) 702
Figure 27-7. MAL TX Descriptor Control/Status Field 703
Figure 27-8. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets) 706
Figure 27-9. MAL RX Descriptor Control/Status Field 709
Figure 27-10. Wake-Up Packet Format 711
Figure 27-11. Control Packet Format 712
Figure 27-12. Integrated Flow Control Mechanism 714
Figure 27-13. Pause Operation State Machine 715
Figure 27-14. VLAN Tagged Packet Format 716
Figure 27-15. Tag Control Information Field Structure 716
Figure 27-16. Receive Address Recognition Flowchart 719
Figure 27-17. Ethernet Address Filter Operation 720
Figure 27-18. Mode Register 0 (EMACx_MR0) 723
Figure 27-19. Mode Register 1 (EMACx_MR1) 724
Figure 27-20. Transmit Mode Register 0 (EMACx_TMR0) 725

Figure 27-21. Transmit Mode Register 1 (EMACx_TMR1) 726
Figure 27-22. Receive Mode Register (EMACx_RMR) 726
Figure 27-23. Interrupt Status Register (EMACx_ISR) 728
Figure 27-24. Interrupt Status Enable Register (EMACx_ISER) 730
Figure 27-25. Individual Address High Register (EMACx_IAHR) 732
Figure 27-26. Individual Address Low Register (EMACx_IALR) 732
Figure 27-27. VLAN TPID Register (EMACx_VTPID) 733
Figure 27-28. VLAN TCI Register (EMACx_VTCI) 733
Figure 27-29. Pause Timer Register (EMACx_PTR) 733
Figure 27-30. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4) 734
Figure 27-31. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4) 734
Figure 27-32. Last Source Address High Register (EMACx_LSAH) 734
Figure 27-33. Last Source Address Low Register (EMACx_LSAL) 734
Figure 27-34. inter-Packet Gap Value Register (EMACx_IPGVR) 735
Figure 27-35. STA Control Register (EMACx_STACR) 735
Figure 27-36. Transmit Request Threshold Register (EMACx_TRTR) 737
Figure 27-37. Receive Low/High Water Mark Register (EMACx_RWMR) 738
Figure 27-38. Number of Octets Transmitted (EMACx_OCTX) 738
Figure 27-39. Number of Octets Received (EMACx_OCRX) 738
Figure 27-40. EMAC-MAL Communication Phases 739
Figure 28-1. UART Receiver Buffer Registers (UARTx_RBR) 747
Figure 28-2. UART Transmitter Holding Registers (UARTx_THR) 747
Figure 28-3. UART Interrupt Enable Registers (UARTx_IER) 747
Figure 28-4. UART Interrupt Identification Registers (UARTx_IIR) 748
Figure 28-5. UART FIFO Control Registers (UARTx_FCR) 749
Figure 28-6. UART Line Control Registers (UARTx_LCR) 750
Figure 28-7. UART Modem Control Registers (UARTx_MCR) 750
Figure 28-8. UART Line Status Registers (UARTx_LSR) 751
Figure 28-9. UART Modem Status Registers (UARTx_MSR) 752
Figure 28-10. UART Scratchpad Registers (UARTx_SCR) 753
Figure 28-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM) 753
Figure 28-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL) 753
Figure 28-13. UART Configuration Register 0 (SDR0_UART0) 754
Figure 28-14. UART Configuration Register 1 (SDR0_UART1) 755
Figure 28-15. UART Configuration Register 2 (SDR0_UART2) 756
Figure 28-16. UART Configuration Register 3 (SDR0_UART3) 756
Figure 29-1. SPI Functional Diagram 770
Figure 29-2. SPI Receive Data Register (SPI0_RxD) 772
Figure 29-3. SPI Transmit Data Register (SPI0_TxD) 772
Figure 29-4. SPI Control Register (SPI0_CR) 773
Figure 29-5. SPI Clock Divisor Modulus Register (SPI0_CDM) 773

Figure 29-6. SPI Status Register (SPI0_SR) 774
Figure 29-7. SPI Mode Register (SPI0_MODE) 774
Figure 30-1. 7-Bit Addressing 776
Figure 30-2. 10-Bit Addressing 776
Figure 30-3. IICx Master Data Buffer (IICx_MDBUF) 778
Figure 30-4. IICx Slave Data Buffer (IICx_SDBUF) 779
Figure 30-5. IICx Low Master Address Register (IICx_LMADR) 779
Figure 30-6. IICx High Master Address Register (IICx_HMADR) 780
Figure 30-7. IICx Control Register (IICx_CNTL) 780
Figure 30-8. IICx Mode Control Register (IICx_MDCNTL) 782
Figure 30-9. IICx Status Register (IICx_STS) 783
Figure 30-10. IICx Extended Status Register (IICx_EXTSTS) 784
Figure 30-11. IICx Low Slave Address Register (IICx_LSADR) 786
Figure 30-12. IICx High Slave Address Register (IICx_HSADR) 786
Figure 30-13. IICx Clock Divide Register (IICx_CLKDIV) 787
Figure 30-14. IICx Interrupt Mask Register (IICx_INTRMSK) 788
Figure 30-15. IICx Transfer Count Register (IICx_XFRCNT) 788
Figure 30-16. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) 789
Figure 30-17. IICx Direct Control Register (IICx_DIRECTCNTL) 790
Figure 30-18. IICx Interrupt Register (IICx_INTR) 791
Figure 31-1. GPIO Dataflow and Configuration Registers 796
Figure 31-2. GPIO Output Register (GPIOx_OR) 799
Figure 31-3. GPIO Three-State Register (GPIOx_TCR) 799
Figure 31-4. GPIO Open Drain Register (GPIOx_ODR) 801
Figure 31-5. GPIO Input Register (GPIOx_IR) 801
Figure 32-1. USB2PHY0 Control Register (SDR0_USB2PHY0CR) 810
Figure 32-2. USB2 Host Status Register (SDR0_USB2H0ST) 811
Figure 32-3. USB2 Host Control Register (SDR0_USB2H0CR) 812
Figure 32-4. USB2 Device Control Register (SDR0_USB2D0CR) 813
Figure 32-5. USB 2.0 PHY Configuration 813
Figure 32-6. USB 2.0 Clocking 815
Figure 32-7. Version/Capability Length Register (EHCI0_HCCAPBASE) 820
Figure 32-8. Structural Parameters Register (EHCI0_HCSPARAMS) 821
Figure 32-9. Capability Parameters Register (EHCI0_HCCPARAMS) 822
Figure 32-10. USB Command Register (EHCI0_USBCMD) 823
Figure 32-11. USB Status Register (EHCI0_USBSTS) 825
Figure 32-12. USB Interrupt Enable Register (EHCI0_USBINTR) 827
Figure 32-13. EHCI Frame Index Register (EHCI0_FRINDEX) 828
Figure 32-14. Control Data Structure Segment Register (EHCI0_CRTLSEG) 829
Figure 32-15. Periodic Frame List Base Address Register (EHCI0_PRDLISTB) 829
Figure 32-16. Current Asynchronous List Address Register (EHCI0_ASYNCLSTA) 830

Figure 32-17. Configure Flag Register (EHCI0_CFGFLAG) 830
Figure 32-18. Programmable Microframe Base Value Register (EHCI0_INSNREG0) 835
Figure 32-19. Programmable Packet Buffer Threshold Register (EHCI0_INSNREG1) 836
Figure 32-20. Programmable Packet Buffer Depth Register (EHCI0_INSNREG2) 837
Figure 32-21. Break Memory Transfer Register (EHCI0_INSNREG3) 837
Figure 32-22. Debug Only Register (EHCI0_INSNREG4) 838
Figure 32-23. Revision Register (OHCI0_HCREV) 838
Figure 32-24. Control Register (OHCI0_HCCTRL) 839
Figure 32-25. Command Status Register (OHCI0_HCCMDSTS) 841
Figure 32-26. Interrupt Status Register (OHCI0_HCINTSTS) 842
Figure 32-27. Interrupt Enable Register (OHCI0_HCINTE) 843
Figure 32-28. Interrupt Disable Register (OHCI0_HCINTDIS) 844
Figure 32-29. Host Controller Communications Area Register (OHCI0_HCHCCA) 845
Figure 32-30. Periodic Current Endpoint Descriptor Register (OHCI0_HCPCED) 845
Figure 32-31. Control Head Endpoint Descriptor Register (OHCI0_HCCHED) 846
Figure 32-32. Control Current Endpoint Descriptor Register (OHCI0_HCCTRLCED) 846
Figure 32-33. Bulk Head Endpoint Descriptor Register (OHCI0_HCBULKHED) 847
Figure 32-34. Bulk Current Endpoint Descriptor Register (OHCI0_HCBULKCED) 847
Figure 32-35. Done Head Transfer Descriptor Register (OHCI0_HCDHEAD) 848
Figure 32-36. Frame Interval Register (OHCI0_HCFMINT) 849
Figure 32-37. Frame Remaining Register (OHCI0_HCFMREM) 850
Figure 32-38. Frame Number Register (OHCI0_HCFMNUM) 850
Figure 32-39. eriodic Start Register (OHCI0_HCPRDSTRT) 851
Figure 32-40. Low Speed Threahold Register (OHCI0_HCLSTHRES) 851
Figure 32-41. Root Hub Descriptor A Register (OHCI0_HCRHDESCA) 852
Figure 32-42. Root Hub Descriptor B Register (OHCI0_HCRHDESCB) 854
Figure 32-43. Root Hub Status Register (OHCI0_HCRHSTS) 855
Figure 32-44. Root Hub Port Status Register (OHCI0_HCPRSTS) 856
Figure 32-45. Bridge Control DCR (USB2HMDCR0_BCNTL) 858
Figure 32-46. Bridge Status DCR (USB2HMDCR0_BSTAT) 859
Figure 32-47. Bridge OPB Latency DCR (USB2HMDCR0_LAT) 860
Figure 32-48. Bridge Revision ID DCR (USB2HMDCR0_REVID) 861
Figure 32-49. Bridge Control DCR (USB2HSDCR0_BCNTL) 862
Figure 32-50. Bridge Revision ID DCR (USB2HSDCR0_REVID) 863
Figure 32-51. Typical USB 2.0 Device Interface System Application Block Diagram 864
Figure 32-52. USB 2.0 Device Interface System Block Diagram 865
Figure 32-53. Interrupts for Endpoints 0 and IN 1–3 Register (USB2D0_INTRIN) 867
Figure 32-54. Power Management Register (USB2D0_POWER) 868
Figure 32-55. Function Address Register (USB2D0_FADDR) 869
Figure 32-56. Interrupt Enable for USB2D0_INTRIN Register (USB2D0_INTRINE) 870
Figure 32-57. Interrupts for OUT Endpoints 1 to 3 Register (USB2D0_INTROUT) 870

Figure 32-58. Interrupt enable for USB2D0_INTRUSB Register (USB2D0_INTRUSBE) 871
Figure 32-59. Interrupts for Common USB Register (USB2D0_INTRUSB) 872
Figure 32-60. Interrupt Enable for USB2D0_INTROUT Register (USB2D0_INTROUTE) 873
Figure 32-61. Enable the USB 2.0 Test Modes Register (USB2D0_TSTMODE) 874
Figure 32-62. Endpoint Status/Control Selection Index Register (USB2D0_INDEX) 875
Figure 32-63. Frame Number Register (USB2D0_FRAME) 875
Figure 32-64. Endpoint 0 Control Status Register (USB2D0_INCSR0) 877
Figure 32-65. Control/Status for IN Endpoint Register (USB2D0_INCSR) 878
Figure 32-66. Maximum Packet Size for IN Endpoint Register (USB2D0_INMAXP) 880
Figure 32-67. Control/Status for OUT Endpoint Register (USB2D0_OUTCSR) 881
Figure 32-68. Maximum Packet Size for OUT Endpoint Register (USB2D0_OUTMAXP) 883
Figure 32-69. Number of Received Bytes in Endpoint 0 FIFO Register (USB2D0_OUTCOUNT0) 884
Figure 32-70. Number of Bytes in OUT Endpoint FIFO Register (USB2D0_OUTCOUNT) 884

Tables

Table 2-1. PPC440EPx/GRx PLB4 Master and Slave Assignments 58
Table 2-2. PPC440EPx/GRx PLB3 Master and Slave Assignments 58
Table 2-3. Registers Controlling PLB4 Master Priority Assignments 59
Table 2-4. Registers Controlling PLB3 Master Priority Assignments 60
Table 2-5. PLB4 Arbiters 0 and 1 Registers 69
Table 2-6. Master Priority Orders 71
Table 2-7. PLB Segment Access 76
Table 2-8. PLB4 to PLB3 Bridge Registers 76
Table 2-9. PLB3 to PLB4 Bridge Registers 84
Table 2-10. PLB4 to OPB Bridge Registers 91
Table 2-11. PLB3 Arbiter 0 Registers 96
Table 2-12. PLB3 to OPB Bridge Registers 99
Table 2-13. Master Assignments 103
Table 2-14. OPB Arbiter Registers 103
Table 2-15. OPB-to-PLB4 Bridge Registers 105
Table 3-1. System Memory Address Map 112
Table 3-2. DCR Address Map (4KB of Device Configuration Registers) 114
Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) 115
Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) 125
Table 4-1. Data Operand Definitions 140
Table 4-2. Alignment Effects for Storage Access Instructions 140
Table 4-3. Invalid Operation Exception Categories 141
Table 4-4. Floating-Point Single Format 145
Table 4-6. Format Fields 146
Table 4-7. IEEE 754 Floating-Point Fields 146
Table 4-5. Floating-Point Double Format 146
Table 4-8. Rounding Modes 151
Table 4-9. IEEE 64-bit Execution Model 152
Table 4-10. Interpretation of the G, R, and X Bits 152
Table 4-11. Location of the Guard, Round, and Sticky Bits in the IEEE Execution Model 153
Table 4-12. Multiply-Add 64-bit Execution Model 154
Table 4-13. Location of Guard, Round, and Sticky Bits in the Multiply-Add Execution Model 154
Table 4-14. Floating-Point Load Instructions 157
Table 4-15. Floating-Point Store Instructions 158
Table 4-16. Floating-Point Move Instructions 159
Table 4-17. Floating-Point Elementary Arithmetic Instructions 159
Table 4-18. Floating-Point Multiply-Add Instructions 160
Table 4-19. Floating-Point Rounding and Conversion Instructions 160
Table 4-20. Comparison Sets 160
Table 4-21. Floating-Point Compare and Select Instructions 161

Table 4-22. Floating-Point Status and Control Register Instructions 161
Table 7-1. Reset Values of Registers and Other PPC440 Core Facilities 172
Table 7-2. SDR Configuration Registers 176
Table 8-1. Bootstrap Register Bit Fields 187
Table 8-2. Bootstrap Pins 189
Table 8-3. Bootstrap Configurations 191
Table 8-4. Serial ROM Memory Map 195
Table 8-5. System DCR Register 196
Table 8-6. IIC Bootstrap Controller Registers 197
Table 9-1. UIC0 Interrupt Assignments 207
Table 9-2. UIC1 Interrupt Assignments 208
Table 9-3. UIC2 Interrupt Assignments 209
Table 9-4. UIC Device Control Registers 210
Table 11-1. Invalid Operation Exception Categories 250
Table 11-2. MSR[FE0, FE1] Modes 252
Table 11-3. Invalid Operation Exceptions 254
Table 11-4. QNaN Result 259
Table 11-5. FPSCR[FPRF] Result Flags 259
Table 11-6. Bit Encodings for a CR Field 261
Table 13-1. GPT Registers 267
Table 14-1. PPC440EPx/GRx Clocking Modes 275
Table 14-2. Clock Frequencies and M multiplier 276
Table 14-3. System PLL Configuration Using PLL Local Feedback (CPR0_PLLC0[SEL] = 000) 276
Table 14-4. System PLL Configuration Using CPU Feedback (CPR0_PLLC0[SEL] = 001) 276
Table 14-6. CPR0_PLLC0[TUNE] Bit Settings 277
Table 14-5. System PLL Configuration Using PerClk Feedback (CPR0_PLLC0[SEL] = 101) 277
Table 14-7. SDR0_CP440[Nto1] Settings 278
Table 14-8. System Clock Rules 279
Table 14-9. Equations to Determine VCO, CPU, PLB Frequency 279
Table 14-10. Clock Ratio Listing With PLL Local Feedback, SysClk = 33.3MHz 280
Table 14-11. Clock Ratio Listing With CPU Feedback, SysClk = 33.3MHz 281
Table 14-12. Clock Ratio Listing With PerClk feedback, SysClk = 33.3MHz 282
Table 14-13. CPU:PLB Ratio 282
Table 14-14. Example Synchronous PCI Clock Frequencies in Asynchronous Mode 283
Table 14-15. PPC440EPx/GRx Clocking Control Register Access 284
Table 14-16. PPC440EPx/GRx Clocking Control Registers 284
Table 15-1. CPM Registers 291
Table 17-1. IPsec ESP Header Processing 302
Table 17-2. IPsec AH Processing 302
Table 17-3. Security Function Registers 307
Table 17-4. Basic Operation Decoding 341

Table 17-5. Protocol Operation Decoding 341
Table 18-1. KASUMI Flow Control Signals 393
Table 18-2. KASUMI Registers 393
Table 19-1. Default Controller Memory Address Mapping 407
Table 19-2. DDR address fields and controlling parameters 408
Table 19-3. Alternate Controller Memory Address Mapping 408
Table 19-4. DDR Address Parameters for Common Organizations 409
Table 19-5. Row/Column Organization of Common DDR Devices 409
Table 19-6. DDR SDRAM Controller DCR Addresses 410
Table 19-7. Parameter Size to Mapping Condition 411
Table 19-8. DDR SDRAM Parameter Catagories 411
Table 19-9. ECC Control Parameter Settings 435
Table 19-10. ECC Error Parameter Description 436
Table 19-11. 64-Bit ECC Syndrome Codes 438
Table 19-12. Parameter XOR_CHECK_BITS Mapping 439
Table 19-13. Delay Parameters 440
Table 19-14. Read Capture Signal Descriptions 441
Table 20-1. SRAM Registers 447
Table 21-1. PowerPC, CoreConnect PLB, and PCI Address Bit-Naming Conventions 456
Table 21-2. PowerPC, CoreConnect PLB, and PCI Data Bus Bit-Naming Conventions 457
Table 21-3. PLB Address Map 459
Table 21-4. PCI Memory Address Map 461
Table 21-5. Transaction Mapping: PLB —> PCI 463
Table 21-6. Transaction Mapping: PCI → PLB 466
Table 21-7. Collision Resolution 470
Table 21-8. Directly Accessed MMIO Registers 471
Table 21-9. PCI Configuration Address and Data Registers 471
Table 21-10. PCI Configuration Register Addresses 472
Table 21-11. PLB Unsupported Transfer Types 494
Table 21-12. Address Map Register Values 499
Table 22-1. EBC Signal Usage 556
Table 22-2. Effect of Driver Enable Programming on EBC Signal States 557
Table 22-3. External Master Arbitration. 569
Table 22-4. Signal States During Hold Acknowledge (HoldAck=1) 570
Table 22-5. EBC DCR Addresses 574
Table 22-6. EBC Configuration and Status Registers 575
Table 23-1. Strap Register Configuration for Booting from NAND Flash 588
Table 23-2. Address/Commands Sequence Auto-Read Mode 589
Table 23-3. Page Read Commands for an Auto-Read Cycle 590
Table 23-4. Page Address for an Auto-Read Cycle 590
Table 23-5. Address Driven with Auto-Read Mode Enabled 590

Table 23-6. ECC Code Assignment Table 592
Table 23-7. Hamming ECC Syndromes 593
Table 23-8. NAND Flash Controller Memory Map 600
Table 23-9. Data Input Ordering to Read Data on EBC 602
Table 23-10. Write Data to Output Ordering 602
Table 24-1. DMA to PLB 4 Channel Assignments 609
Table 24-2. DMA to PLB4 Controller Configuration and Status Registers 611
Table 24-3. DMA to PLB 4 Transfer Priorities 619
Table 24-4. Address Alignment Requirements 620
Table 24-5. DMA to PLB 4 Registers Loaded from Scatter/Gather Descriptor 622
Table 24-6. DMA to PLB 3 Channel Assignments 625
Table 24-7. DMA to PLB3 Controller External I/Os 626
Table 24-8. DMA to PLB 3 Controller Configuration and Status Registers 629
Table 24-9. DMA to PLB 3 Transfer Priorities 636
Table 24-10. Address Alignment Requirements 637
Table 24-11. Scatter/Gather Descriptor Table 638
Table 24-12. Bit Fields in the Scatter/Gather Descriptor Table 639
Table 24-13. DMA Registers Loaded from Scatter/Gather Descriptor Table 639
Table 25-1. MAL0 Channel Assignment 648
Table 25-2. TX Status Control Field 660
Table 25-3. RX Status Control Field 661
Table 25-4. MAL DCR Summary 668
Table 26-1. Ethernet Combinations 681
Table 26-2. Register Address List 684
Table 26-3. RGMII Bridge PHY Interface Signals 688
Table 26-4. RGMII Bridge Registers 689
Table 27-1. EMAC Status and Packet Reject Counter Registers 698
Table 27-2. FCS/SA Enable - Possible Configurations 707
Table 27-3. FCS/Pad Enable - Possible Configurations 707
Table 27-4. FCS/VLAN Tag Enable - Possible Configurations 708
Table 27-5. In Range Length Error Behavior for Various Packet Lengths 710
Table 27-6. EMAC0 Register Summary 720
Table 27-7. EMAC1 Register Summary 721
Table 28-1. Baud Rate Settings 745
Table 28-2. UART Configuration Registers 746
Table 28-3. Interrupt Priority Level 748
Table 28-4. Divisor Latch Settings for Certain Baud Rates 754
Table 28-5. DMA to PLB3 Channel Assignments 759
Table 28-6. UART 0 Transmitter DMA Mode Register Field Settings 761
Table 28-7. UART 1 Transmitter DMA Mode Register Field Settings 762
Table 28-8. UART2 Transmitter DMA Mode Register Field Settings 763

Table 28-9. UART3 Transmitter DMA Mode Register Field Settings 764
Table 28-10. UART0 Receiver DMA Mode Register Field Settings 765
Table 28-11. UART1 Receiver DMA Mode Register Field Settings 766
Table 28-12. UART2 Receiver DMA Mode Register Field Settings 767
Table 28-13. UART3 Receiver DMA Mode Register Field Settings 768
Table 29-1. SPI Registers 771
Table 30-1. IIC Registers 777
Table 30-2. IIC Response to IICx_CNTL Field Settings 781
Table 30-3. IICx_STS[ERR, PT] Decoding 783
Table 30-4. IICx Clock Divide Programming 787
Table 31-1. GPIO Register Summary 797
Table 31-2. GPIO Output Signal Selection 800
Table 31-3. GPIO Three-State Selection 800
Table 31-4. GPIO0_ODR Control Settings 801
Table 31-5. GPIO Alternate Input Signal Selection 802
Table 31-6. Alternate 1 Configuration for GPIO00–GPIO15 803
Table 31-7. Alternate 1 Configuration for GPIO16–GPIO31 804
Table 31-8. Alternate 1 Configuration for GPIO32–GPIO47 805
Table 31-9. Alternate 1 Configuration for GPIO48 805
Table 31-10. Alternate 2 Configuration for GPIO00–GPIO05 806
Table 31-11. Alternate 2 Configuration for GPIO27–GPIO31 806
Table 31-12. Alternate 2 Configuration for GPIO32–GPIO47 807
Table 31-13. Alternate 2 Configuration for GPIO48 807
Table 31-14. Alternate 3 Configuration for GPIO34–GPIO37 807
Table 32-1. USB Configuration Registers 810
Table 32-2. Clock Source Settings 815
Table 32-3. USB Host Registers 818
Table 32-4. Port Status and Control Register (EHCI0_PORTSC) 831
Table 32-5. USB Device Registers 866
Table 32-6. Indexed Registers 876
Table 32-7. FIFO Address Assignment 885
Table 34-1. Instruction Categories 891
Table 34-2. Operator Precedence 894
Table 34-3. fres Operation with Special Operand Values 917
Table 34-4. frsqrte Operation with Special Operand Values 919
Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers 948

About This Book

This user's manual describes the integrated peripherals, their control registers, status registers, configuration options and programming information for the AMCC PowerPC 440EPx/GRx (PPC440EPX/GRx).

Who Should Use This Book

This book is for system hardware and software developers who need to understand the PPC440EPx/GRx Embedded Processor.

How to Use This Book

This book describes the PPC440EPx/GRx device architecture (including instructions and registers), processor core functions, system operations, internal bus functions, and external interfaces. The book is organized as follows:

- Part I Introduction
  - *Overview* on page 55
  - *On-Chip Buses* on page 57
- Part II PPC440EPx/GRx RISC Processor
  - *Programming Model* on page 111
  - *FPU Programming Model* on page 139
  - *Instruction and Data Caches* on page 163
  - *Memory Management* on page 165
- Part III PPC440EPx/GRx System Operations
  - *Reset and Initialization* on page 169
  - *Bootstrap Controller* on page 187
  - *Universal Interrupt Controller* on page 205
  - *Interrupts and Exceptions* on page 247
  - *Floating Point Unit Interrupts and Exceptions* on page 249
  - *Timer Facilities* on page 263
  - *General Purpose Timers* on page 265
  - *Clocking* on page 273
  - *Clock and Power Management* on page 291
  - *Debug Facilities* on page 297
  - *Security Function* on page 299
- Part IV PPC440EPx/GRx Peripheral Functions and Interfaces
  - *DDR SDRAM Controller* on page 405
  - *Internal SRAM Controller* on page 447
  - *Peripheral Component Interconnect (PCI) Interface* on page 455
  - *External Bus Controller* on page 555
  - *NAND Flash Controller* on page 583
  - *Direct Memory Access Controllers* on page 609
  - *Memory Access Layer* on page 645
  - *EMAC to PHY Interface Bridges* on page 681
  - *Ethernet Media Access Controllers* on page 693
  - *Serial Port Operations* on page 743
  - *Serial Peripheral Interface* on page 769
  - *IIC Bus Interface* on page 775
  - *GPIO Operations* on page 795

- *Universal Serial Bus Interfaces* on page 809

- Part V Reference
  - *Instruction Set* on page 889
  - *Register Summary* on page 947
  - *Signal Summary* on page 971

This book contains the following appendixes:
- *Floating Point Instruction Summary* on page 973

Conventions

The following is a list of notational conventions frequently used in this manual.

$\overline{\text{ActiveLow}}$	An overbar indicates an active-low signal.
n	A decimal number
0x*n*	A hexadecimal number
0b*n*	A binary number
=	Assignment
$\wedge$	AND logical operator
$\neg$	NOT logical operator
$\vee$	OR logical operator
$\oplus$	Exclusive-OR (XOR) logical operator
+	Twos complement addition
–	Twos complement subtraction, unary minus
$\times$	Multiplication
$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
||	Concatenation
$=, \neq$	Equal, not equal relations
<, >	Signed comparison relations
$<^u, >^u$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
do	Do loop. "to" and "by" clauses specify incrementing an iteration variable; "while" and "until" clauses specify terminating conditions. Indenting indicates the scope of a loop.
leave	Leave innermost do loop or do loop specified in a leave statement.
FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
REG_b	A bit in a named register

$REG_{b:b}$	A range of bits in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register
REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, . . .
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
CR_{FLD}	The field in the condition register pointed to by a field of an instruction.
$c_{0:3}$	A 4-bit object used to store condition results in compare instructions.
^{n}b	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
CEIL(x)	Least integer $\ge$ x.
EXTS(x)	The result of extending *x* on the left with sign bits.
PC	Program counter.
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies a location in main storage.
EA_b	A bit in an effective address.
$EA_{b:b}$	A range of bits in an effective address.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.

Part I. Introduction

1. Overview

The AMCC PowerPC 440EPx and 440GRx 32-bit RISC embedded processors, are pin compatible closely related system-on-a-chip (SOC) designs containing the PowerPC 440 processor (PPC440 CPU) and a number of integrated controllers including Gigabit EMAC(s), PCI bridge, DDR controller, security engine and external bus controller (EBC). This user's manual contains programming and configuration descriptions of the integrated controllers.

Since the PPC440EPx and PPC440GRx are closely related designs, they are identified separately only where differences exist. Throughout the user's manual, PPC440EPx/GRx is used where text is common to both.

Note:

1. Refer to the "PowerPC 440 Processor User's Manual" for the 440 processor programming model and the descriptions of the caches, MMU, debug features, processor timer facility (watchdog/fixed interval timer/programmable interval timer), special purpose registers (SPR) and interrupts.
2. Block diagrams, detail feature lists, part numbers and PVR numbers are provided in the *PowerPC 440EPx Embedded Processor Data Sheet* and *PowerPC 440GRx Embedded Processor Data Sheet.*

1.1 PPC440EPx and PPC440GRx Differences

Differences between the processors are limited to the FPU, USB and the processor version register number (PVR).

PPC440EPx specific features:

- FPU
- USB Host
- USB Device

PPC440GRx specific features:

- No FPU support
- No USB support

2. On-Chip Buses

The on-chip bus structure, which consists of the processor local bus (PLB), on-chip peripheral bus (OPB), and device control register (DCR) bus, provides a connection among all the masters and slaves on the chip buses. PPC440EPx/GRx implements two processor local buses (128-bit PLB4 and 64-bit PLB3) and three identical on-chip peripheral buses. The block diagram in the *PowerPC 440EPx Embedded Processor Data Sheet and PowerPC 440GRx Embedded Processor Data Sheet* illustrates the on-chip bus structure of the PPC440EPx/GRx.

PLB4 and PLB3 are high performance buses used to access memory through bus interface units.Transfer of data between these two buses is enabled via PLB4 to PLB3 and PLB3 to PLB4 bridges. The PLB master and slave assignments for the PPC440EPx/GRx are listed in *PLB Master and Slave Assignments* on page 58. While some high performance peripherals such as PLB4 to PLB3 bridge, the memory access layer, DDR SDRAM and a DMA controller are attached to PLB4, others such as the PCI bridge, PLB3 to PLB4 bridge and another DMA controller are attached to PLB3. See *Processor Local Bus* on page 57.

Lower performance peripherals (such as serial ports, the Ethernet controller, the external bus controller, general purpose timers, the USB interface, general purpose I/O interface, and IIC controllers) are attached to the OPB. A bridge between the PLB and OPB enables data transfers between PLB masters and OPB slaves. See *On-Chip Peripheral Bus* on page 102.

The DCR bus is used primarily to access status and control registers of the various PLB and OPB masters and slaves. The DCR bus off-loads status and control read and write transfers from the PLB. PPC440EPx/GRx also contains peripherals such as serial port, ethernet controllers and others that use memory mapped registers which are accessed using load/store instructions. See *Device Control Register (DCR) Bus* on page 107.

Note: The term *processor local bus* or PLB in this chapter is used to represent both PLB4 and PLB3 bus unless otherwise mentioned.

2.1 Processor Local Bus

The processor local bus is a high-performance on-chip bus. The PLB supports read and write data transfers between master and slave devices equipped with a PLB interface and connected through PLB signals.

Each PLB master is attached to the PLB through separate address, read data and write data buses, and transfer qualifier signals. PLB slaves are attached to the PLB through shared, but decoupled, address, read data and write data buses, and transfer control and status signals for each data bus.

Access to the PLB is granted through a central arbitration mechanism that enables masters to compete for bus ownership. This arbitration mechanism provides for fixed and fair priority schemes.

Timing for all PLB signals is provided by a clock source that is shared by all PLB masters and slaves.

2.1.1 PLB Features

- Overlapping of read and write transfers allows two concurrent data transfers for maximum bus utilization
- Decoupled address and data buses support split-bus transaction capability for improved bandwidth
- Address pipelining reduces overall bus latency by allowing the latency associated with a new request to be overlapped with an ongoing data transfer in the same direction
- Late master request abort capability reduces latency associated with aborted requests
- Four levels of request priority for each master allow PLB implementations with various arbitration schemes
- Byte-enable capability allows for unaligned transfers and odd-byte transfers.

- Support for fixed length burst transfers
- Guarded and unguarded memory transfers allow a slave device to enable or disable the prefetching of instructions or data
- DMA buffered, peripheral to memory, memory to peripheral, and DMA memory to memory operations are supported

2.1.2 PLB Master and Slave Assignments

Table 2-1 and *Table 2-2* list the PLB4 and the PLB3 masters and slaves provided in the PPC440EPx/GRx.

Table 2-1. PPC440EPx/GRx PLB4 Master and Slave Assignments

PLB Agent	PLB Masters and Slaves	Master/Slave No.
Processor core instruction cache unit (ICU)	Master	0
Processor core data cache read unit (DCU)	Master	1
Processor core data cache write unit (DCU)	Master	2
DMA2P40 controller	Master	3
PLB3 to PLB4 bridge	Master	4
OPB to PLB4 bridge	Master	5
Memory Access Layer (MAL)	Master	6
Security Engine	Master	7
DDR2 SRAM controller	Slave	0, 0
PLB4 to OPB Bridge 0 *(PPC440EPx only)*	Slave	0, 1
PLB4 to OPB Bridge 1 *(PPC440EPx only)*	Slave	0, 2
Security engine	Slave	0, 3
Internal SRAM controller	Slave	0, 4
Kasumi engine	Slave	0, 5
PLB4 to PLB3 bridge	Slave	1, 0

Table 2-2. PPC440EPx/GRx PLB3 Master and Slave Assignments

PLB Agent	PLB Masters and Slaves	Master/Slave No.
PLB4 to PLB3 bridge	Master	0
PCI bridge	Master	2
DMA2P30 controller	Master	3
External bus controller	Master	5
PLB3 to OPB bridge	Slave	0
External bus controller	Slave	1
PCI bridge	Slave	2
PLB3 to PLB4 bridge	Slave	3

2.1.3 PLB Master Priority Assignment

Each PLB master can be programmed to use one of four priority levels during PLB transfers, enabling the system designer to tune PLB transfer priorities to the requirements of a particular application. For example, if an application always requires DMA PCI to SDRAM transfers to have the lowest latency, the DMA PCI master can be programmed to the highest PLB master priority. This causes the PLB arbiter to grant DMA PCI access requests before granting the access requests of any other master.

A register associated with each master controls the priority of that master. *Table 2-3* and *Table 2-4* list the PLB4 and PLB3 masters and the register fields controlling the priority of the masters. Priorities range from 0b00 (lowest) to 0b11 (highest).

Table 2-3. Registers Controlling PLB4 Master Priority Assignments

Master ID	Description	Register Field	Comments
0	Processor core instruction cache unit (ICU)	SDR0_AMP0[AICURP] SDR0_AMP0[ICURP] SDR0_CP440[IRF] SDR0_CP440[IRT] SDR0_CP440[IRS]	Alternate ICU Read Priority ICU Read Priority IcuRdFetch priority setting IcuRdTouch priority setting IcuRdSpec priority setting
1	Processor core data cache read unit (DCU)	SDR0_AMP0[ADCURP] SDR0_AMP0[DCURP] SDR0_CP440[DRU] SDR0_CP440[DRT] SDR0_CP440[DRNC] SDR0_CP440[DRLC]	Alternate DCU Read Priority DCU Read Priority DcuRdUrgent priority setting DcuRdTouch priority setting DcuRdNonCache priority DcuRdLdCache priority setting
2	Processor core data cache write unit (DCU)	SDR0_AMP0[ADCUWP] SDR0_AMP0[DCUWP] SDR0_CP440[DWF] SDR0_CP440[DWS] SDR0_CP440[DWU]	Alternate DCU Write Priority DCU Write Priority DcuWrFlush priority setting DcuWrStore priority setting DcuWrUrgent priority setting
3	DMA2P40 controller	DMA2P40_CR0,1,2,3[CP] SDR0_AMP0[ADMAP4P] SDR0_AMP0[DMAP4P]	Alternate DMA to PLB4 Priority DMA to PLB4 Priority
4	PLB3 to PLB4 bridge	P3P4BI0_CFG[PRI] SDR0_AMP0[AP3P4P] SDR0_AMP0[P3P4P]	Alternate PLB3 to PLB4 Priority PLB3 to PLB4 Priority
5	OPB to PLB4 bridge *(PPC440EPx only)*	OPB2PLB40_BCTRL[PRI] SDR0_AMP0[AOPB2P4P] SDR0_AMP0[OPB2P4P]	Alternate OPB to PLB4 Priority OPB to PLB4 Priority
6	MAL	MAL0_CFG[RPP] MAL0_CFG[WPP]	Read PLB4 priority Write PLB4 Priority
7	Security Engine	CRYP0_DMA_CFG[PLBP] SDR0_AMP1[ACRYP] SDR0_AMP1[CRYP]	Alternate Security Enginel Priority Security Engine Priority

Table 2-4. Registers Controlling PLB3 Master Priority Assignments

Master ID	Description	Register Field	Comments
0	PLB4 to PLB3 bridge	P4P3BO0_CFG[PRI] SDR0_AMP1[AP4P3P] SDR0_AMP1[P4P3P]	Alternate PLB4 to PLB3 bridge priority PLB4 to PLB3 bridge priority
2	PCI bridge	SDR0_AMP1[APCIP] SDR0_AMP1[PCIP]	
3	DMA2P30 controller	DMA2P30_CR0[CP] DMA2P30_CR1[CP] DMA2P30_CR2[CP] DMA2P30_CR3[CP] SDR0_AMP1[ADMA1P] SDR0_AMP1[DMA1P]	Unique priorities can be assigned to each DMA channel
5	External bus controller	EBC0_CFG[EMPL] EBC0_CFG[EMPH] SDR0_AMP1[AEBCP] SDR0_AMP1[EBCP]	Which field sets external master priority depends upon the setting of the HoldPri signal.
6:7	Reserved		

Note: PLB master priority assignments are application-dependent, and must be considered carefully in order to prevent potential lockouts of lower priority masters. For most applications, assigning a priority of 0b10 to each master is a useful starting point. See *PLB4 Arbiter Control Register (PLB4An_ACR)* on page 70 and *PLB3 Arbiter 0 Control Register (PLB3A0_ACR)* on page 98 for information about programming the PLB4A0_ACR and PLB3A0_ACR to control PLB priority mode and priority order. The PLB0_ACR must be set to "fair" mode arbitration. This helps to prevent various lockout scenarios.

2.1.3.1 Alternate PLB4 Master Priority Register 0 (SDR0_AMP0)

SDR0_AMP0 is a 32-bit read/write register containing alternative PLB4 master priority setting.

Reset value = 0 for all fields.

Figure 2-1. Alternate PLB4 Master Priority Register (SDR0_AMP0)

0:1	AICURP	Alternate ICU Read Priority 00 Lowest 01 10 11 Highest	
2:3	ADCURP	Alternate DCU Read Priority 00 Lowest 01 10 11 Highest	
4:5	ADCUWP	Alternate DCU Write Priority 00 Lowest 01 10 11 Highest	

6:7	ADMA4P	Alternate DMA2P40 Priority 00 Lowest 01 10 11 Highest	
8:9	AP3P4P	Alternate PLB3 to PLB4 Bridge Priority 00 Lowest 01 10 11 Highest	
10:11	A0P4P	Alternate OPB0 to PLB4 Bridge Priority 00 Lowest 01 10 11 Highest	*(PPC440EPx only)*
12:13		Reserved	
14:15	ACRYPP	Alternate CRYP Priority 00 Lowest 01 10 11 Highest	
16	ICURP	ICU Read Priority 0 ICU read controls own priority 1 ICU read uses alternate priority	
17	DCURP	DCU Read Priority 0 DCU read controls own priority 1 DCU read uses alternate priority	
18	DCUWP	DCU Write Priority 0 DCU write controls own priority 1 DCU write uses alternate priority	
19	DMA4P	DMA2P40 Priority 0 DMA to PLB4 controls own priority 1 DMA to PLB4 uses alternate priority	
20	P3P4P	PLB3 to PLB4 Priority 0 PLB3 to PLB4 controls own priority 1 PLB3 to PLB4 uses alternate priority	
21	OP4P	OPB2PLB40 Priority 0 OPB2PLB40 controls own priority 1 OPB2PLB40 uses alternate priority	*(PPC440EPx only)*
22		Reserved	
23	CRYPP	CRYP Priority 0 CRYP controls own priority 1 CRYP uses alternate priority	
24:31		Reserved	

2.1.3.2 Alternate PLB3 Master Priority Register 1 (SDR0_AMP1)

SDR0_AMP1 is a a 32-bit read/write register containing alternative PLB3 master priority setting.

Reset value = 0 for all fields.

Figure 2-2. Alternate PLB3 Master Priority Register 1 (SDR0_AMP1)

0:1	AP4P3P	Alternate PLB4 to PLB3 Bridge Priority 00 Lowest 01 10 11 Highest	
2:3		Reserved	
4:5	APCIP	Alternate PCI Priority 00 Lowest 01 10 11 Highest	
6:7	ADMA3P	Alternate DMA2P30 Priority 00 Lowest 01 10 11 Highest	
8:9		Reserved	
10:11	AEBCP	Alternate EBC Priority 00 Lowest 01 10 11 Highest	
12:15		Reserved	
16	P4P3P	PLB4 to PLB3 Bridge Priority 0 PLB4 to PLB3 bridge controls own priority 1 PLB4 to PLB3 bridge uses alternate priority	
17		Reserved	
18	PCIP	PCI Priority 0 PCI controls own priority 1 PCI uses alternate priority	
19	DMA3P	DMA2P30 Priority 0 DMA2P30 controls own priority 1 DMA2P30 uses alternate priority	
20		Reserved	
21	EBCP	EBC Priority 0 EBC controls own priority 1 EBC uses alternate priority	
22:31		Reserved	

2.1.3.3 PPC440 CPU Control Register (SDR0_CP440)

SDR0_CP440 is a 32-bit read/write register that controls processor core priorities

Reset value = see individual field definitions.

Figure 2-3. PPC440 CPU Register (SDR0_CP440)

0:1		Reserved	Reset value = 0b00.
2:3	RL	Boot ROM Location 00 EBC 01 PCI 10 NDFC 11 Reserved	Specifies the boot source. Reset value = SDR0_SDSTP1[RL]
4:5	DRU	DcuRdUrgent 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with an urgent state in which two or more read data cache operations are pending, waiting for the previous request to be serviced. PLB master priority is updated to this value when in urgent state regardless of instruction type. Reset value = 0b11.
6:7	DRT	DcuRdTouch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with dcbt instructions except when in urgent state. Reset value = 0b10.
8:9	DRNC	DcuRdNonCache 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with non-cacheable load instructions except when in urgent state. Reset value = 0b10.
10:11	DRLC	DcuRdLdCache 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with cacheable load instructions except when in urgent state. Reset value = 0b10.
12:13	DWF	DcuWrFlush 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with flush instructions except when in urgent state. Reset value = 0b10.
14:15	DWS	DcuWrStore 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with store instructions except when in urgent state. Reset value = 0b10.
16:17	DWU	DcuWrUrgent 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with an urgent state in which two or more write data cache operations are pending, waiting for the previous request to be serviced. PLB master priority is updated to this value when in urgent state regardless of instruction type. Reset value = 0b11.
18:19	IRF	IcuRdFetch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with non-speculative ICU accesses Reset value = 0b10.

20:21	IRT	IcuRdTouch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with icbt instructions. Reset value = 0b10.
22:23	IRS	IcuRdSpec 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with speculative ICU accesses. Reset value = 0b10.
24:29		Reserved	Reset value = 0b00.
30	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	Reset value = SDR0_SDSTP1[Nto1]
31		Reserved	Reset value = 0b00.

2.1.3.4 PLB4 Master Interrupt Request Registers 0 and 1 (SDR0_MIRQ0, SDR0_MIRQ1)

SDR0_MIRQ0 and SDR0_MIRQ1 are a 32-bit read/write registers that contains status of interrupt request generated by PLB4 slaves. A PLB4 slave can assert an interrupt request to a PLB4 master whenever the slave encounters an event which it deems important to the master. This event can be because of an operation that was initiated by the master or not. Once the slave makes an interrupt request, the request remains asserted until cleared by a subsequent access to the slave. This access can be via the PLB or DCR interface.

Software can examine the error address register and error status register of the slave making the interrupt request to determine for which transfer the interrupt request applies.

Reset value = 0 for all fields.

Figure 2-4. Master Interrupt Request Register 0 (SDR0_MIRQ0)

0:1		Reserved	
2	M0DDR	ICU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
3	M0P4O0	ICU read interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
4	M0P4O1	ICU read interrupt request from PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
5:6		Reserved	
7	M0P4P3	ICU read interrupt request from PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	

8:9		Reserved	
10	M1DDR	DCU read interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
11	M1P4O0	DCU read interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
12	M1P4PO1	DCU read interrupt request from PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
13:14		Reserved	
15	M1P4P3	DCU read interrupt request from PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
16:17		Reserved	
18	M2DDR	DCU write interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
19	M2P4O0	DCU write interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
20	M2P4O1	DCU write interrupt request from PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
21:22		Reserved	
23	M2P4P3	DCU write interrupt request from PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
24:25		Reserved	
26	M3DDR	DMA2P4 interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
27	M3P4O0	DMA2P4 interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
28	M3P4O1	DMA2P4 interrupt requestfrom PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
29:30		Reserved	
31	M3P4P3	DMA2P4 interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	

Figure 2-5. Master Interrupt Request Register 1 (SDR0_MIRQ1)

0:1		Reserved	
2	M4DDR	P3P4BI0 bridge interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
3	M4P4O0	P3P4BI0 bridge interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
4	M4P4O1	P3P4BI0 bridge interrupt request from PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
5:6		Reserved	
7	M4P4P3	P3P4BI0 bridge interrupt request from PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
8:9		Reserved	
10	M5DDR	OPB2PLB40 interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
11	M5P4O0	OPB2PLB40 interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
12	M5P4O1	OPB2PLB40 interrupt request from PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
13:14		Reserved	
15	M5P4P3	PLB42OPB0 interrupt request from PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
16:17		Reserved	
18	M6DDR	MAL0 interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
19	M6P4O0	MAL0 interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
20	M6P4O1	MAL0 interrupt request from PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
21:22		Reserved	

23	M6P4P3	MAL0 interrupt request from PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	
24:25		Reserved	
26	M7DDR	CRYP0 interrupt request from DDR SDRAM 0 Interrupt request inactive 1 Interrupt request active	
27	M7P4O0	CRYP0 interrupt request from PLB4 to OPB0 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
28	M7P4O1	CRYP0 interrupt requestfrom PLB4 to OPB1 bridge 0 Interrupt request inactive 1 Interrupt request active	*(PPC440EPx only)*
29:30		Reserved	
31	M7P4P3	CRYP0 interrupt requestfrom PLB4 to PLB3 bridge 0 Interrupt request inactive 1 Interrupt request active	

2.1.3.5 PLB Slave Address Pipeline Register (SDR0_SLPIPE0)

SDR0_SLPIPE0 is a 32-bit read/write register that controls PLB4 and PLB3 slave address pipeline.

Figure 2-6. Slave Address Pipeline Register (SDR0_SLPIPE0)

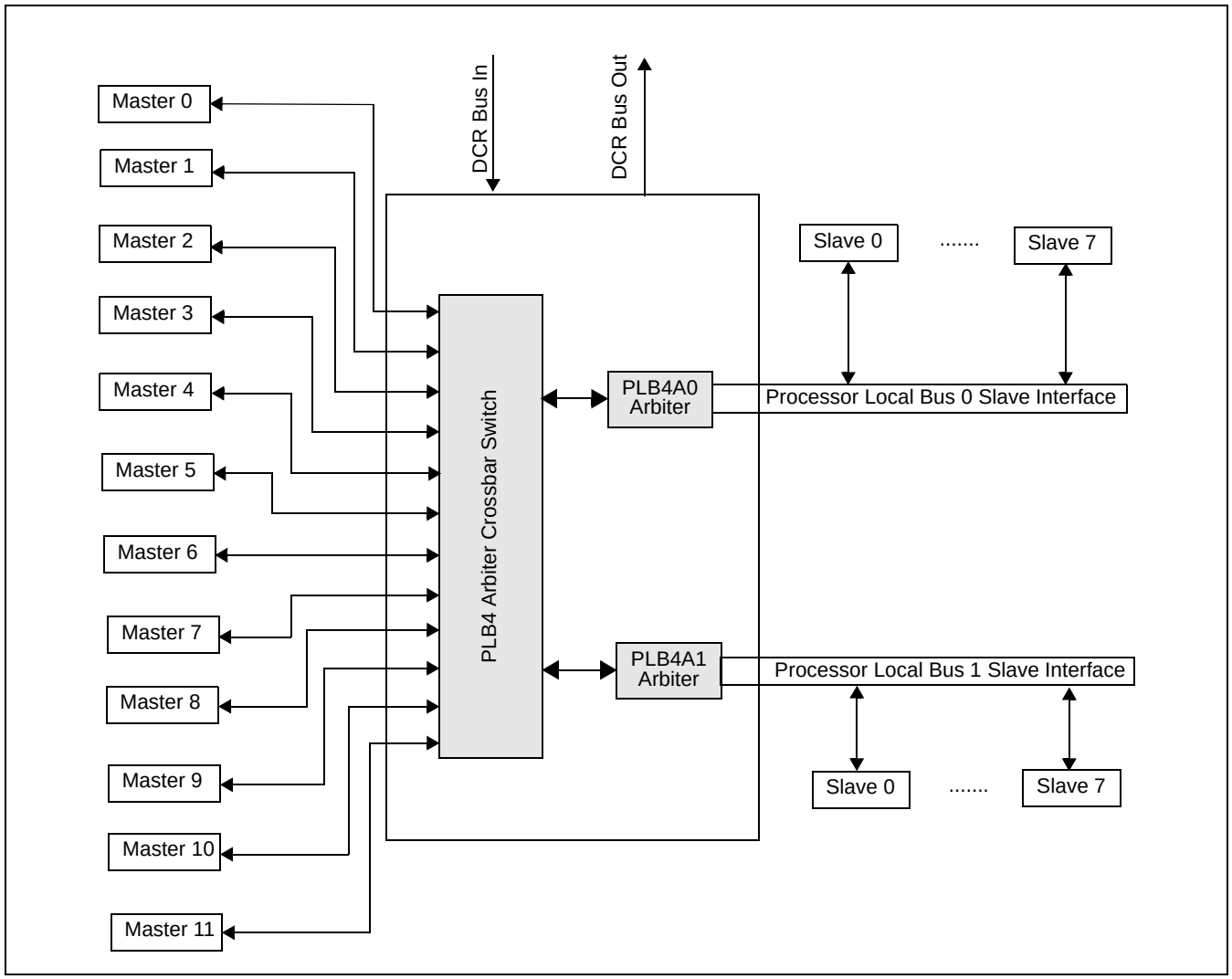
0		Reserved	
1	DAP	DDR Address Pipeline 0 DDR address pipeline disabled 1 DDR address pipeline enabled	
2	P4P3AP	PLB4 to PLB3 Address Pipeline 0 PLB4 to PLB3 address pipeline disabled 1 PLB4 to PLB3 address pipeline enabled	
3	P4OAP	PLB4 to OPB Address Pipeline 0 PLB4 to OPB address pipeline disabled 1 PLB4 to OPB address pipeline enabled	Addresses both PLB4 to OPB 0 bridge and PLB4 to OPB 1 bridge.*(PPC440EPx only)*
4:27		Reserved	
28	P3OAP	PLB3 to OPB Address Pipeline 0 PLB3 to OPB address pipeline disabled 1 PLB3 to OPB address pipeline enabled	
29	EAP	EBC Address Pipeline 0 EBC address pipeline disabled 1 EBC address pipeline enabled	
30	PAP	PCI Address Pipeline 0 PCI address pipeline disabled 1 PCI address pipeline enabled	
31	P3P4AP	PLB3 to PLB4 Address Pipeline 0 PL3 to PLB4 address pipeline disabled 1 PLB3 to PLB4 address pipeline enabled	

2.1.4 PLB4 Arbiter

This is the high performance128-bit version crossbar switch supporting two processor local bus arbiters implementing the three cycle acknowledge timing protocol. It is a soft core consisting of a 12-master to 2-PLB slave segments crossbar switch including for each PLB segment: a bus arbitration control unit, a watchdog timer, address path, write data path, and read data path units.

Figure 2-7 shows how the PLB crossbar arbiter is interconnected.

Figure 2-7. PLB Crossbar Arbiter Interconnection



As shown in *Figure 2-7*, the on-chip bus structure provides a link between PLB bus masters such as the processor core, DMA controller, OPB to PLB bridge, and other PLB master devices and PLB bus slaves such as the memory controller, PLB to OPB bridge, and other PLB slave devices, residing on either the PLB0 slave interface or the PLB1 slave interface.

The processor local bus (PLB) is the high performance bus used to access memory through the bus interface units. This PLB is implemented with a crossbar switch allowing separate PLB master devices to independently access slave devices attached to PLB 0 slave interface and PLB1 slave Interface concurrently for improved PLB bus performance.

The device control register (DCR) bus is used primarily for accessing status and control registers. It is meant to off-load the PLB from the lower performance status and control read and write transfers. The DCR bus architecture allows data transfers among peripherals to occur independently from, and concurrent with, data transfers between the PLB masters and PLB slaves.

2.1.4.1 PLB4 Arbiters 0 and 1 Registers

PLB4 arbiter registers listed in *Table 2-5* are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-5. PLB4 Arbiters 0 and 1 Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB4A0_REVID	PLB4 Crossbar ID/Revision Register	0x0080	R	69
PLB4A0_ACR	PLB4A0 Arbiter Control Register	0x0081	R/W	70
PLB4A0_ESRL	PLB4A0 Error Status Register Low	0x0082	R/Clear	73
	Reserved	0x0083		
PLB4A0_EARL	PLB4A0 Error Address Register Low	0x0084	R	75
PLB4A0_EARH	PLB4A0 Error Address Register High	0x0085	R	75
PLB4A0_ESRL*	PLB4A0 Error Status Register Low (*reserved for diagnostic use only)	0x0086	Set(W)	73
PLB4A0_ESRH*	PLB4A0 Error Status Register High (*reserved for diagnostic use only)	0x0087	Set(W)	
PLB4A0_CCR	PLB4 Crossbar Control Register	0x0088	R/W	
PLB4A1_ACR	PLB4A1 Arbiter Control Register	0x0089	R/W	70
PLB4A1_ESRL	PLB4A1 Error Status Register Low	0x008A	R/Clear	73
	Reserved	0x008B		
PLB4A1_EARL	PLB4A1 Error Address Register Low	0x008C	R	75
PLB4A1_EARH	PLB4A1 Error Address Register High	0x008D	R	75
PLB4A1_ESRL*	PLB4A1 Error Status Register Low (*reserved for diagnostic use only)	0x008E	Set(W)	73
PLB4A1_ESRH*	PLB4A1 Error Status Register High (*reserved for diagnostic use only)	0x008F	Set(W)	

2.1.4.2 PLB4 Arbiter Revision ID Register (PLB4A0_REVID)

PLB4A0_REVID is a 32-bit read-only register that contains the revision ID of the PLB4 arbiter. The contents of the register can be accessed by using the move from device control register (mfdcr) instruction. The PLB4A0_REVID register can be accessed with CPU_dcrAddr(6:9) = 0x2. The register is not affected by reset.

Figure 2-8. PLB4A0 Arbiter 0 Revision ID Register (PLB4A0_REVID)

0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

2.1.4.3 PLB4 Arbiter Control Register (PLB4An_ACR)

PLB4An_ACR is a 32-bit register which controls the modes of operation for the arbiter. The priority mode, high bus utilization, read pipeline enable, and write pipeline enable are contained in this register. At reset, Fair priority mode for all priority levels is selected, high bus utilization is enabled, and two deep read and write pipelining are enabled. PLB4An_ACR[0:31] = 0xDB000000.

Figure 2-9. PLB4 Arbiter Control Register (PLB4An_ACR)

0:3	PPM	PLB Priority Mode PPM0 0 Priority level 00 Fixed priority. 1 Priority level 00 Fair priority. PPM1 0 Priority level 01 Fixed priority 1 Priority level 01 Fair priority PPM2 - Reserved (always zero) PPM3 0 Priority level 11 Fixed priority 1 Priority level 11 Fair priority	
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write pipeline disabled 1 2 Deep write pipe	
8:31		Reserved	

The following sections provide details on the effect of the bit settings in PLB4An_ACR.

PPM Field

During the bus arbitration cycle, the bus arbitration control unit uses the Mn_priority0:1 signals to determine which master will be granted the bus. The priority inputs of masters with their respective Mn_request signal asserted are used in determine the highest request priority. In addition, the core supports the fixed priority and the fair priority scheme to handle "tie" situations (that is, situations when two or more masters request the bus simultaneously while presenting the same level of request priority). The selection of the priority mode during tie situations is controlled by these bits.

Under the fixed priority scheme, each bus master is assigned a unique priority level as shown below. Note that only three distinct levels of priority are supported by the Crossbar Arbiter. A master requesting on Priority Level 2 is treated by the Crossbar Arbiter as if it had requested on Priority Level 1. *Table 2-6* shows all five bus masters in the order of priority:

Table 2-6. Master Priority Orders

Highest Priority	Decreasing Priority ----->			Lowest Priority
For Priority Level 3 (highest):				
M0	M1	M2	M3	M4
For Priority Level 2:				
Merged with Priority Level 1				
For Priority Level 1:				
M0	M1	M2	M3	M4
For Priority Level 0 (lowest):				
M0	M1	M2	M3	M4

Similarly, under the fair priority scheme, each bus master is assigned a unique priority level as shown above. However, once one master at a particular Mn_priority0:1 level is granted the bus, all other masters requesting the bus at that priority level at that time that continue to present their request at that same priority level will eventually be granted the bus before any other master (including the one originally granted the bus) is subsequently granted the bus at that priority level.

Once the last master that had not been granted the bus which is still requesting the bus on that priority level is granted the bus (or if no masters that had not been granted the bus continue to request the bus at that priority level), arbitration for the next master will be open to all requesting masters at that priority level again using the original priority above. If two masters request the bus simultaneously and continuously at the same priority level each master will be granted the bus 50% of the time at that priority level. If three masters request the bus simultaneously and continuously at the same priority level, each master will be granted the bus 33% of the time at that priority level. If four masters request the bus simultaneously and continuously at the same priority level, each master will be granted the bus 25% of the time at that priority level, etc.

This assumes all masters assert and continue to assert the same request priority. Note that a "qualified" request is a Mn_Request addressed to this PLB segment which is not otherwise blocked (for example, by the same master having a "like transaction" in progress on the other PLB segment).

The detailed manner in which this fairness algorithm works is as follows:

- For the PLB4An arbiter, there is one "fairness request" latch for each of the 12 masters on each of the three distinct priority levels (for a total of 36 latches for the PLB4An arbiter).
- If fairness on a particular level is *not* enabled, then the latches for that level are ignored (treated as if they were set to 0).
- If fairness on a particular level is enabled and *all* latches on that level are set to 0 (the initial condition), then on the arbitration cycle, arbitration on that level occurs among *all* masters with valid "qualified" requests active on that arbitration level. In the same clock cycle, the "fairness request" latches corresponding to each master with a valid "qualified" request on that arbitration level are set to one by the next clock rising edge.
- If fairness on a particular level is enabled and at least one latch on that level is set to one, then arbitration on that level occurs *only* among those masters with a valid "qualified" request on that arbitration level whose "fairness request" latch is set to one by the next clock rising edge.
- On *any* clock cycle, if a master stops presenting a valid "qualified" request on a particular arbitration level, then that master's "fairness request" latch is cleared to zero by the next clock rising edge.

- On *any* clock cycle, if arbAddrSelRegn becomes active (indicating that Master n has "won" the current arbitration), then each "fairness request" latch associated with Master n is cleared to 0 by the next rising clock edge.

In this manner, all sustained "qualified" requests which have been latched in the "fairness request" latches on the same arbitration level must be satisfied once for PLB4An before any master's request on the same level may be granted a second time on PLB4An. Note that a "qualified" request on a particular arbitration level may be removed without being granted as a result of the master dynamically changing the requested arbitration level or by the master aborting the request.

In this example, when arbAddrSelRegn is a 1 this indicates that master n has been latched as winner of the current arbitration on PLB0 (until terminated after a slave responds with addrack or rearbitrate, or the arbiter's watchdog timer issues a timeout).

HBU Bit

This bit enables a feature that will ensure both the read and write data buses are always busy if there are pending master requests for both buses. An additional requirement is that there are no bus lock requests pending or bus locked transactions in progress. These conditions will temporarily disable this feature. When this bit is 1 the arbiter will promote a lower priority request to the current active request on the address bus under the following conditions, (there are two cases, one for each data bus):

- If the read data bus is busy with a transfer and a secondary read request has been address acknowledged (addrAcked) and there are additional read request(s) pending, and the next highest request in the arbitration queue is a read request. Then a lower priority master write request will be promoted to the current active write bus request on the address bus if the write data bus is idle.
- A similar case also exists for the write bus being busy with both a primary request active and a secondary request address acknowledged (addrAcked) and the next highest request in the arbitration queue is a write request also. If a lower priority read request is pending and the read data bus is idle then the lower priority read request will be promoted to the active request on the address bus.

If read and write pipelining are disabled this feature will have no affect on arbiter operation. The default mode after reset is enabled. Setting this bit to 0 will disable this feature. It is recommended this bit be cleared immediately after reset.

RDP Field

These bits control the depth of read pipelining. That is the number of outstanding read requests broadcast to the slaves. When PLB4An_ACR[RDP] = 01 the arbiter will default to a two deep read pipeline after reset. This allows the arbiter to broadcast a primary and secondary read to the slaves. When PLB4An_ACR[RDP] = 10 the arbiter is capable of generating a primary, secondary, and third read to the slaves. When PLB4A0_ACR[RDP] = 11 the arbiter is capable of generating a primary, secondary, third, and fourth read to the slaves. Clearing these bits disables read pipelining and only primary read transfers are broadcast. In this case PLB0_SAValid will never be asserted for a read request. The arbiter has eight separate slave address acknowledge signals, Sl_PLB0_addrAck0:7. This allows the arbiter to determine which slave is responding to the PLB0_SAValid signal assertion. The arbiter also has eight separate read primary signals, PLB0_rdPrim0:7, which should be connected to the corresponding slave. Consequently upon completion of the primary transfer the arbiter will notify the appropriate slave that its pipelined transfer is now the primary and may begin driving the data bus and controls two cycles following the assertion of PLB0_rdPrimn.

WRP Field

This bit controls the depth of write pipelining. That is the number of outstanding write requests broadcast to the slaves. When PLB0_ACR[WRP] = 1 the arbiter defaults to a two deep write pipeline after reset. This allows the arbiter to broadcast a primary and secondary write to the slaves. When PLB0_ACR[WRP] = 0 write pipelining will be disabled and only primary write transfers will be broadcast. In this case PLB0_SAValid will never be asserted for a write request.

2.1.4.4 PLB4 Error Status Register Low (PLB4An_ESRL)

The PLB4An_ESRL register described in the following figure identifies time-out errors on PLB4 bus transfers, the master initiating the transfer, and the type of transfer. Each PLB4An_ESRL[PTEn] field (n in the field is the master ID) can be locked by the master. Once locked, PLB4An_ESRL[PTEn] fields cannot be updated if a subsequent error occurs until the corresponding PLB4An_ESRL[FLKn] field is cleared. To clear a PLB4An_ESRL field, write 1 to the field. Writing 0 to a PLB4A0_ESRL field does not affect the field.

The PLB4A0_ESRL register can be accessed with DCR address = 0x82 (read/clear) and DCR address = 0x86 (set). The PLB4A1_ESRL can be accessed with DCR address = 0x8A (read/clear) and DCR address= 0x8E (set). At reset, all bits in the PLB4An_ESRL are loaded with zeroes. The registers at DCR = 0x86 and 0x8E are for diagnostic purposes only.

Figure 2-10. PLB4 Error Status Register Low (PLB4An_ESRL)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 - Processor core instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB4A0_ESR Field Lock 0 Master 0 PLB4A0_ESR field is unlocked 1 Master 0 PLB4A0_ESR field is locked	
3	ALK0	Master 0 PLB4A0_EAR Address Lock 0 Master 0 PLB4A0_EAR is unlocked 1 Master 0 PLB4A0_EAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 - Processor core data cache read unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1PLB4A0_ESR Field Lock 0 Master 1 PLB4A0_ESR field is unlocked 1 Master 1 PLB4A0_ESR field is locked	
7	ALK1	Master 1 PLB4A0_EAR Address Lock 0 Master 1 PLB4A0_EAR is unlocked 1 Master 1 PLB4A0_EAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 - Processor core data cache write unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB4A0_ESR Field Lock 0 Master 2 PLB4A0_ESR field is unlocked 1 Master 2 PLB4A0_ESR field is locked	
11	ALK2	Master 2 PLB4A0_EAR Address Lock 0 Master 2 PLB4A0_EAR is unlocked 1 Master 2 PLB4A0_EAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 - DMA2P40

13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB4A0_ESR Field Lock 0 Master 3 PLB4A0_ESR field is unlocked 1 Master 3 PLB4A0_ESR field is locked	
15	ALK3	Master 3 PLB4A0_EAR Address Lock 0 Master 3 PLB4A0_EAR is unlocked 1 Master 3 PLB4A0_EAR is locked	
16	PTE4	Master 4 PLB Timeout Error Status 0 No Master 4 timeout error 1 Master 4 timeout error	Master 4 - PLB3 to PLB 4 bridge
17	R/W4	Master 4 Read/Write Status 0 Master 4 error operation was a write 1 Master 4 error operation was a read	
18	FLK4	Master 4 PLB4A0_ESR Field Lock 0 Master 4 PLB4A0_ESR field is unlocked 1 Master 4 PLB4A0_ESR field is locked	
19	ALK4	Master 4 PLB4A0_EAR Address Lock 0 Master 4 PLB4A0_EAR is unlocked 1 Master 4 PLB4A0_EAR is locked	
20	PTE5	Master 5 PLB Timeout Error Status 0 No Master 5 timeout error 1 Master 5 timeout error	Master 5 - OPB to PLB4 Bridge 1 *(PPC440EPx only)*
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	*(PPC440EPx only)*
22	FLK5	Master 5 PLB4A0_ESR Field Lock 0 Master 5 PLB4A0_ESR field is unlocked 1 Master 5 PLB4A0_ESR field is locked	*(PPC440EPx only)*
23	ALK5	Master 5 PLB4A0_EAR Address Lock 0 Master 5 PLB4A0_EAR is unlocked 1 Master 5 PLB4A0_EAR is locked	*(PPC440EPx only)*
24	PTE6	Master 6 PLB Timeout Error Status 0 No Master 6 timeout error 1 Master 6 timeout error	Master 6 - MAL For fields to become locked, an error must occur while MAL has error locking enabled (MAL0_CFG[PLBLE] = 1)
25	R/W6	Master 6 Read/Write Status 0 Master 6 error operation was a write 1 Master 6 error operation was a read	
26	FLK6	Master 6 PLB4A0_ESR Field Lock 0 Master 6 PLB4A0_ESR field is unlocked 1 Master 6 PLB4A0_ESR field is locked	
27	ALK6	Master 6 PLB4A0_EAR Address Lock 0 Master 6 PLB4A0_EAR is unlocked 1 Master 6 PLB4A0_EAR is locked	
28	PTE7	Master 7 PLB Timeout Error Status 0 No Master 7 timeout error 1 Master 7 timeout error	Master 7 - Security Engine
29	R/W7	Master 7 Read/Write Status 0 Master 7 error operation was a write 1 Master 7 error operation was a read	

30	FLK7	Master 7 PLB4A0_ESR Field Lock 0 Master 7 PLB4A0_ESR field is unlocked 1 Master 7 PLB4A0_ESR field is locked	
31	ALK7	Master 7 PLB4A0_EAR Address Lock 0 Master 7 PLB4A0_EAR is unlocked 1 Master 7 PLB4A0_EAR is locked	

2.1.4.5 PLB4 Error Address Register Low (PLB4An_EARL)

The read-only PLB4An_EARL register contains the lower 32 bits of the address of the access on which a bus time-out error occurred. The PLB4An_EARL can be locked by the master. Once locked, the PLB4An_EARL cannot be updated, if a subsequent error occurs, until all PLB4An_ESRL[FLCKn] and PLB4An_ESRH[FLCKn] fields are cleared (n is the master ID).

The PLB4An_EARL is not affected by Reset.

Figure 2-11. PLB4 Error Address Register Low (PLB4An_EARL)

0:31		PLB4An lower address of bus timeout error	

2.1.4.6 PLB4 Error Address Register High (PLB4An_EARH)

The read-only PLB4An_EARH register contains the upper 32 bits of the address of the access on which a bus time-out error occurred. The PLB4An_EARH can be locked by the master. Once locked, the PLB4An_EARH cannot be updated, if a subsequent error occurs, until all PLB4An_ESRL[FLCKn] and PLB4An_ESRH[FLCKn] fields are cleared (n is the master ID).

The PLB4An_EARH is not affected by Reset.

Figure 2-12. PLB4 Error Address Register High (PLB4An_EARH)

0:31		PLB4An upper address of bus timeout error

2.1.4.7 PLB4 Crossbar Control Register (PLB4A0_CCR)

The PLB Crossbar Control Register (PLB4A0_CCR) controls the modes of operation for the crossbar switch. The PLB Segmentation Address Decode selection is contained in this register. At reset PLB4A0_CCR[PSA] is loaded with the value of PGM_PLBSegAddr0:3. At reset PLB4A0_CCR[PLB3] is loaded with the value of PGM_PLB3TO0:11.

Figure 2-13. PLB4 Crossbar Control Register (PLB4A0_CCR)			
0:3	PSA	PLB Segmentation Upper Address Control See *Table 2-7*	These bits, in combination with Mn_SegABus0:3, determine which slave PLB segment a Master's Mn_request is routed to by the crossbar switch. These bits must be set to 0x8.
4:15	PLB3	PLB3 TimeOut Mode Enabled For each PLB3 bit: 0 Crossbar arbiter's PLB4 timeout mode enabled for this master port 1 Crossbar arbiter's PLB3 timeout mode enabled for this master port	The bits in PLB4A0_CCR[PLB3] correspond to Master Ports 0:5 respectively. These bits must be set to 0.
16:31		Reserved	

Each bit of the PSA field determines whether the corresponding bit of Mn_SegABus0:3 will be used in determining whether to access the PLB4A0 or the PLB4A1 slave segment. If *all* of the Mn_SegABus0:3 bits for which the corresponding bit of PSA is a 1 are also 1s, then PLB4A1 is accessed. Otherwise PLB4A0 is accessed. *Table 2-7* shows how PSA bits are used with Mn_SegAbus0:3 to map master accesses to either PLB4A0 or PLB4A1.

Table 2-7. PLB Segment Access

PSA[0:3]	Mn_segAbus0:3		Access PLB Segment
	From	To	
1000	0000	0111	PLB4A0
1000	1000	1111	PLB4A1

2.1.5 PLB4 to PLB3 Bridge Registers (Bridge Out 0)

The PLB4 to PLB3 bridge registers are DCRs that are accessed using the mfdcr and mtdcr instructions.

Table 2-8. PLB4 to PLB3 Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
P4P3BO0_BESR0	PLB4 to PLB3 Bridge Error Status Register 0 Master Devices 0,1,2,3	0x0020	R/Clear	77
P4P3BO0_BEARL	PLB4 to PLB3 Bridge Error Address Register Low	0x0022	R/Clear	77
P4P3BO0_BEARH	PLB4 to PLB3 Bridge Error Address Register High	0x0023	R/Clear	77
P4P3BO0_BESR1	PLB4 to PLB3 Bridge Error Status Register 1 Master Devices 4,5,6,7	0x0024	R/Clear	80
P4P3BO0_CFG	PLB4 to PLB3 Bridge Configuration Register	0x0026	R/Clear/Set	82
P4P3BO0_PICR	PLB4 to PLB3 Bridge Priority Incrementation Counter Register	0x0027	R/Clear/Set	83
P4P3BO0_PEIR	PLB4 to PLB3 Bridge Parity Error Interrupt Register	0x0028	R/Clear/Set	84
P4P3BO0_REVID	PLB4 to PLB3 Bridge Revision ID Register	0x002A	R/Clear	84

2.1.5.1 PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL)

P4P3BO0_BEARL, described in the following figure, is a read-only register. As part of its error reporting, the PLB4 to PLB3 bridge loads the P4P3BO0_BEARL with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB4 to PLB3 bridge accepted the PLB transfer, the P4P3BO0_BEARL will lock, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P4P3BO0_BESRn). Once locked, the P4P3BO0_BEARL cannot be overwritten until all the P4P3BO0_BESRn address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-14. PLB4 to PLB3 Bridge Error Address Register Low (P4P3BO0_BEARL)

0:31		Lower address of bus timeout error

2.1.5.2 PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH)

P4P3BO0_BEARH described in the following figure is a read-only register. As part of its error reporting, the PLB4-to-PLB3 bridge loads the P4P3BO0_BEARH register with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB4-to-PLB3 Bridge accepted the PLB transfer, the P4P3BO0_BEARH locks, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P4P3BO0_BESR). Once locked, the P4P3BO0_BEARH cannot be overwritten until all the P4P3BO0_BESRn address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-15. PLB4 to PLB3 Bridge Error Address Register High (P4P3BO0_BEARH)

0:31		Upper address of bus timeout error	

2.1.5.3 PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0)

In addition to driving the current PLB master's error input signal the PLB4 to PLB3 bridge also records the error information into the appropriate master's P4P3BO0_BESR0 field (provided that the register is not already locked by a previous error that was locked), where it could be optionally locked by the master having its Mn_lockError signal asserted at the time the PLB4 to PLB3 bridge accepted the PLB operation. Parity errors and the WIRQ bit operate independently of the field lock function. Thus, if a master locks its field with the lock error function and receives either a parity error or another write error then these bits will be updated.

The WIRQ bit position, in each of the master's field, is used to drive an interrupt like signal back to the system if a write error condition is encountered. This signal stays asserted until cleared by a DCR software access that resets the WIRQ bit. Note that this bit position does not behave like the PTE and R/W bit positions in that if the register is already locked by a previous error then these bit positions will not get modified from any subsequent errors. However, the WIRQ bit will always get set upon detection of a write error and only get reset by a DCR access to clear this bit. It is not affected by the field lock bit. Its function is independent of the field lock bit. For write parity errors, resetting the PAR bits will reset the interrupt, PPBx_MIRQ for the associated PLB master that encountered the write parity error on its write data bus. The PPBx__ABusParErr and PPBX__rdDBusParErr interrupt bits are both reset in the *PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR)* on page 84

Once locked, a master's P4P3BO0_BESR0 field cannot be overwritten if any subsequent data or time out error occurs, until cleared by using the move to device control register (mtdcr) instruction. To clear either P4P3BO0_BESRn, a 1 must be loaded into those register bits that are to be cleared using the proper DCR access address. Writing a 0 to any bit in either P4P3BO0_BESR will not affect the status of that bit.

The PLB4 to PLB3 bridge contains two P4P3BO0_BESR's: P4P3BO0_BESR0 logs error data for PLB masters 0-3, and P4P3BO0_BESR1 logs error data for PLB masters 4-7.

Figure 2-16. PLB4 to PLB3 Bridge Error Status Register 0 (P4P3BO0_BESR0)

0:1	SSET0	Secondary PLB Bus Slave Error Type for Master 0 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Data error occurred 11 Reserved	Master 0 - Processor core instruction cache unit (ICU)
2	R/W0	Read write status Master 0 0 Error operation is a write 1 Error operation is a read	
3	FLK0	P4P3BO0_BESR field lock Master 0 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
4	ALK0	P4P3BO0_BEAR address lock Master 0 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
6:7	PAR0	Parity Error Type Master 0 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	
8:9	SSET1	Secondary PLB Bus Slave Error Type for Master 1 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 1 - Processor core data cache read unit (DCU)
10	R/W1	Read/write status Master 1 0 Error operation is a write 1 Error operation is a read	
11	FLK1	P4P3BO0_BESR field lock Master 1 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
12	ALK1	P4P3BO0_BEAR address lock Master 1 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
14:15	PAR1	Parity Error Type Master 1 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected from 11) Write bus parity error detected from data supplied	

16:17	SSET2	Secondary PLB Bus Slave Error Type for Master 2 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 2 - Processor core data cache writeunit (DCU)
18	R/W2	Read/write status Master 2 0 Error operation is a write 1 Error operation is a read	
19	FLK2	P4P3BO0_BESR field lock Master 2 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
20	ALK2	P4P3BO0_BEAR address lock Master 2 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
22:23	PAR2	Parity Error Type Master 2 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	
24:25	SSET3	Secondary PLB Bus Slave Error Type for Master 3 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 3 - DMA2P40
26	R/W3	Read/write status Master 3 0 Error operation is a write 1 Error operation is a read	
27	FLK3	P4P3BO0_BESR field lock Master 3 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
28	ALK3	P4P3BO0_BEAR address lock Master 3 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
30:31	PAR3	Parity Error Type Master 3 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	

2.1.5.4 PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1)

Figure 2-17. PLB4 to PLB3 Bridge Error Status Register 1 (P4P3BO0_BESR1)

0:1	SSET4	Secondary PLB Bus Slave Error Type for master 4 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 4 - PLB3 to PLB4 bridge
2	R/W4	Read write status master 4 0 Error operation is a write 1 Error operation is a read	
3	FLK4	P4P3BO0_BESR field lock master 4 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
4	ALK4	P4P3BO0_BEAR address lock master 4 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
5	WIRQ4	Write Error Interrupt master 4 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
6:7	PAR4	Parity Error Type master 5 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	
8:9	SSET5	Secondary PLB Bus Slave Error Type for master 5 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 5 - OPB to PLB4 Bridge *(PPC440EPx only)*
10	R/W5	Read write status master 5 0 Error operation is a write 1 Error operation is a read	*(PPC440EPx only)*
11	FLK5	P4P3BO0_BESR field lock master 5 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	*(PPC440EPx only)*
12	ALK5	P4P3BO0_BEAR address lock master 5 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	*(PPC440EPx only)*
13	WIRQ5	Write Error Interrupt master 5 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	*(PPC440EPx only)*
14:15	PAR5	Parity Error Type master 5 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	*(PPC440EPx only)*

16:17	SSET6	Secondary PLB Bus Slave Error Type for master 6 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 6 - MAL
18	R/W6	Read write status master 6 0 Error operation is a write 1 Error operation is a read	
19	FLK6	P4P3BO0_BESR field lock master 6 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
20	ALK6	P4P3BO0_BEAR address lock master 6 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
21	WIRQ6	Write Error Interrupt master 6 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
22:23	PAR6	Parity Error Type master 6 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	
24:25	SSET7	Secondary PLB Bus Slave Error Type for master 7 00 No error occurred 01 Timeout error occurred (4x mode only) 10 Slave error occurred 11 Reserved	Master 7 - Security function
26	R/W7	Read write status master 7 0 Error operation is a write 1 Error operation is a read	
27	FLK7	P4P3BO0_BESR field lock master 7 0 P4P3BO0_BESR field is unlocked 1 P4P3BO0_BESR field is locked	
28	ALK7	P4P3BO0_BEAR address lock master 7 0 P4P3BO0_BEAR address is unlocked 1 P4P3BO0_BEAR address is locked	
29	WIRQ7	Write Error Interrupt master 7 0) No write error detected - interrupt request is inactive 1) Write error detected - interrupt request is active	
30:31	PAR7	Parity Error Type master 7 00) No parity errors detected 01) Read bus parity error detected from secondary PLB slave device read data 10) Address bus parity error or BE parity error detected 11) Write bus parity error detected from data supplied	

2.1.5.5 PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG)

Figure 2-18. PLB4 to PLB3 Bridge Configuration Register (P4P3BO0_CFG)

0	SSPS	Slave side primary/secondary priority 0 This PLB4-to-PLB3 Bridge Slave side is the secondary Slave side. 1 This PLB4-to-PLB3 Bridge Slave side is the primary Slave side.	
1:2	FIFOE	FIFO Enablement 00 PLB4-to-PLB3 Bridge Address FIFO is disabled. 01 PLB4-to-PLB3 Bridge Acknowledged Address FIFO is set to one level deep. 10 PLB4-to-PLB3 Bridge Acknowledged Address FIFO is set to two levels deep. 11 Reserved	These are read only bits
3	SRE	Secondary Read Enablement 0 0) PLB4-to-PLB3 Bridge secondary read feature is disabled. 1 1) PLB4-to-PLB3 Bridge secondary read feature is enabled.	Read only bit
4	SWE	Secondary Write Enablement 0 PLB4-to-PLB3 Bridge secondary write feature is disabled. 1 PLB4-to-PLB3 Bridge secondary write feature is enabled.	Read only bit
5	PBASS	PLB Bus Architecture for Slave Side 0 PLB4-to-PLB3 Bridge Slave side is connected to a 3.x PLB Bus. 1 PLB4-to-PLB3 Bridge Slave side is connected to a 4.x PLB Bus.	
6	PBAMS	PLB Bus Architecture for Master Side 0 PLB4-to-PLB3 Bridge Master side is connected to a 3.x PLB Bus. 1 PLB4-to-PLB3 Bridge Master side is connected to a 4.x PLB Bus.	
7	ER	Enable Rearbitration 0 PLB4-to-PLB3 Bridge Slave side rearbitration is disabled. 1 PLB4-to-PLB3 Bridge Slave side rearbitration is enabled.	Read only
8:9	PRI	PLB Priority Bits 00 Lowest 01 Next Highest 10 Next Highest 11 Highest	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_PPB4_priority input.
10	PE	Parity Enable 0 Parity checking is disabled 1 Parity checking is enabled on a per side basis depending on whether connected to a 4.x PLB Bus.	During reset this bit is set to the value present on PGM_PPB4_parEnable input.

11	FI	Flush Interrupt Enable 0 Interrupt flushing of writes is disabled and the interrupt will not be forwarded to B-PLB slave via FlushIntComp 1 Interrupt flushing of writes from A-PLB master 'x' is enabled and will be forwarded to B-PLB slave via FlushIntComp	
12:14	FIID	Flush Interrupt ID	Value programmed must match the master ID that the A-PLB bus master 'X' is connected to. (For example, if master 'X' is connected to port 4 of the A-PLB arbiter then the FIID should be programmed to 3'b100)
15:31		Reserved	

2.1.5.6 PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR)

When PPBx_PICR[PICE] = 0, PLB4 to PLB3 master side request priority will not increment and is fixed at the programmed value, set in P4P3BO0_CONFG[8:9].

When P4P3BO0_PICR[PICE] = 1, (this is the default after system reset) if the PLB4-to-PLB3 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. Then, if the PLB4-to-PLB3 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. This continues until 0b11 is reached. For next request, priority value starts out at the original value.

Initial Priority Incrementation Counter Value (1:5 bits): The PLB4-to-PLB3 bridge counts blocks of 8 PLB clocks for each value of the counter. Thus, for a count value of 0b00000 the counter would count 8 PLB clocks and increment its request priority if it has not received an acknowledgement from the target PLB slave device. A count value of 0b00001 would expire after 16 PLB clocks, a value of 0b00010 would expire after 24 PLB clocks, etc.

Figure 2-19. PLB4 to PLB3 Bridge Priority Incrementation Counter Register (P4P3BO0_PICR)

0	PICE	Priority Incrementation Counter Enable 0 Counter is disabled. 1 Counter is enabled.	
1:5	IPICV	Initial Priority Incrementation Counter Value The maximum count value, 5'b11111, is 256 PLB clocks, or 1.92 us at 133MHz.	Value resets to zero. The maximum count value, 5'b11111, is 256 PLB clocks, or 1.92 us at 133MHz.
6:31		Reserved	

2.1.5.7 PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR)

Figure 2-20. PLB4 to PLB3 Bridge Parity Error Interrupt Register (P4P3BO0_PEIR)			
0	ADRPE	A Bus Parity Error Interrupt 0 No Upper address, lower address or byte enable parity errors detected. 1 Parity error detected on either the upper address, lower address of byte enables of requested transfer to PLB4-to-PLB3 Bridge. PLB4-to-PLB3 address parity error interrupt is active.	
1	RDPE	Read Data Bus Parity Error Interrupt 0 No read data bus parity error detected 1 Read data bus parity error detected from secondary PLB Bus slave device data transfer to PLB4-to-PLB3 Bridge. PLB4-to-PLB3 read data parity error interrupt is active.	
2:31		Reserved	

2.1.5.8 PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID)

Figure 2-21. PLB4 to PLB3 Bridge Revision ID Register (P4P3BO0_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL (hard wired to 0x2)
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL (hard wired to 0x1)

2.1.6 PLB3 to PLB4 Bridge Registers (Bridge In 0)

The PLB3 to PLB4 bridge registers are DCRs that are accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-9. PLB3 to PLB4 Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
P3P4BI0_BESR0	PLB3 to PLB4 Bridge Error Status Register 0 Master Devices 0,1,2,3	0x0030	R/Clear	85
P3P4BI0_BEARL	PLB3 to PLB4 Bridge Error Address Register Low	0x0032	R/Clear	85
P3P4BI0_BEARH	PLB3 to PLB4 Bridge Error Address Register High	0x0033	R/Clear	85
P3P4BI0_BESR1	PLB3 to PLB4 Bridge Error Status Register 1 Master Devices 4,5,6,7	0x0034	R/Clear	87
P3P4BI0_CFG	PLB3 to PLB4 Bridge Configuration Register	0x0036	R/Clear/Set	88
P3P4BI0_PICR	PLB3 to PLB4 Bridge Priority Incrementation Counter Register	0x0037	R/Clear/Set	89
P3P4BI0_PEIR	PLB3 to PLB4 Bridge Parity Error Interrupt Register	0x0038	R/Clear/Set	90
P3P4BI0_REVID	PLB3 to PLB4 Bridge Revision ID Register	0x003A	R/Clear	90

2.1.6.1 PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL)

P3P4BI0_BEARL is a read-only register. As part of its error reporting, the PLB3 to PLB4 bridge loads the P3P4BI0_BEARL with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB3 to PLB4 bridge accepted the PLB transfer, the P3P4BI0_BEARL will lock, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P3P4BI0_BESR). Once locked, the P3P4BI0_BEARL cannot be over-written until all the P3P4BI0_BESR address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-22. PLB3 to PLB4 Bridge Error Address Register Low (P3P4BI0_BEARL)

0:31		Lower address of bus timeout error	

2.1.6.2 PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH)

P3P4BI0_BEARH is a read-only register. As part of its error reporting, the PLB3-to-PLB4 bridge loads the P3P4BI0_BEARH register with the error address if the ALCK, (address lock bit), is not already set. If the master's Mn_lockError signal was asserted when the PLB3-to-PLB4 Bridge accepted the PLB transfer, the P3P4BI0_BEARH locks, (if the ALCK bit is not already set either by this master or another master), upon loading the first error address (the master's ALCK bit is set in the P3P4BI0_BESR). Once locked, the P3P4BI0_BEARH cannot be overwritten until all the P3P4BI0_BESR address locking bits, (total of eight, one for each master), are cleared by using the move to device control register (mtdcr) instruction.

Figure 2-23. PLB3 to PLB4 Bridge Error Address Register High (P3P4BI0_BEARH)

0:31		Upper address of bus timeout error	

2.1.6.3 PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0)

In addition to driving the current PLB master's error input signal the PLB3 to PLB4 bridge also records the error information into the appropriate master's P3P4BI0_BESR0 field (provided that the register is not already locked by a previous error that was locked), where it could be optionally locked by the master having its Mn_lockError signal asserted at the time the PLB3 to PLB4 bridge accepted the PLB operation. Parity errors and the WIRQ bit operate independently of the field lock function. Thus, if a master locks its field with the lock error function and receives either a parity error or another write error then these bits will be updated.

The WIRQ bit position, in each of the master's field, is used to drive an interrupt like signal back to the system if a write error condition is encountered. This signal stays asserted until cleared by a DCR software access that resets the WIRQ bit. Note that this bit position does not behave like the PTE and R/W bit positions in that if the register is already locked by a previous error then these bit positions will not get modified from any subsequent errors. However, the WIRQ bit will always get set upon detection of a write error and only get reset by a DCR access to clear this bit. It is not affected by the field lock bit. Its function is independent of the field lock bit. For write parity errors, resetting the PAR bits will reset the interrupt, PPBx_MIRQ for the associated PLB master that encountered the write parity error on its write data bus. The PPBx__ABusParErr and PPBx__rdDBusParErr interrupt bits are both reset in the *PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR)* on page 90

Once locked, a master's P3P4BI0_BESR0 field cannot be overwritten if any subsequent data or time out error occurs, until cleared b using the move to device control register (mtdcr) instruction. To clear either P3P4BI0_BESRn, a 1 must be loaded into those register bits that are to be cleared using the proper DCR access address. Writing a 0 to any bit in either P3P4BI0_BESR will not affect the status of that bit.

The PLB3 to PLB4 bridge contains two P3P4BI0_BESR's: P3P4BI0_BESR0 logs error data for PLB masters 0-3, and P3P4BI0_BESR1 logs error data for PLB masters 4-7.

Figure 2-24. PLB3 to PLB4 Bridge Error Status Register 0 (P3P4BI0_BESR0)

0:1	SSET0	Secondary PLB Bus Slave Error Type for master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred (4x mode only) 10 Master 0 data error occurred 11 Reserved	
2	R/W0	Read write status master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3	FLK0	P3P4BI0_BESR field lock master 0 0 Master 0 P3P4BI0_BESR field is unlocked 1 Master 0 P3P4BI0_BESR field is locked	
4	ALK0	P3P4BI0_BEAR address lock master 0 0 Master 0 P3P4BI0_BEAR address is unlocked 1 Master 0 P3P4BI0_BEAR address is locked	
5	WIRQ0	Write Error Interrupt master 0 0) No write error detected - master 0 interrupt request is inactive 1) Write error detected - master 0 interrupt request is active	
6:7	PAR0	Parity Error Type master 0 00) No parity errors detected for master 0 01) Read bus parity error detected from secondary PLB slave device read data for master 0 10) Address bus parity error or BE parity error detected from master 0 11) Write bus parity error detected from data supplied by master 0	
8:15		Reserved	
16:17	SSET2	Secondary PLB Bus Slave Error Type for master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred (4x mode only) 10 Master 2 slave error occurred 11 Reserved	
18	R/W2	Read/write status master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	P3P4BI0_BESR field lock master 2 0 Master 2 P3P4BI0_BESR field is unlocked 1 Master 2P3P4BI0_BESR field is locked	
20	ALK2	P3P3BO0_BEAR address lock master 2 0 Master 2 P3P4BI0_BEAR address is unlocked 1 Master 2 P3P4BI0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt master 2 0) No write error detected - master 2interrupt request is inactive 1) Write error detected - master 2interrupt request is active	

22:23	PAR2	Parity Error Type master 2 00) No parity errors detected for master 2 01) Read bus parity error detected from secondary PLB slave device read data for master 2 10) Address bus parity error or BE parity error detected from master 2 11) Write bus parity error detected from data supplied by master 2	
24	SSET3	Secondary PLB Bus Slave Error Type for master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred (4x mode only) 10 Master 3 slave error occurred 11 Reserved	
25	R/W3	Read/write status master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
26	FLK3	P3P4BI0_BESR field lock master 3 0 Master 3 P3P4BI0_BESR field is unlocked 1 Master 3 P3P4BI0_BESR field is locked	
27	ALK3	P3P4BI0_BEAR address lock master 3 0 Master 3 P3P4BI0_BEAR address is unlocked 1 Master 3 P3P4BI0_BEAR address is locked	
28	WIRQ3	Write Error Interrupt master 3 0) No write error detected - master 3interrupt request is inactive 1) Write error detected - master 3interrupt request is active	
30:31	PAR3	Parity Error Type master 3 00) No parity errors detected for master 3 01) Read bus parity error detected from secondary PLB slave device read data for master 3 10) Address bus parity error or BE parity error detected from master 3 11) Write bus parity error detected from data supplied by master 3	

2.1.6.4 PLB3 to PLB4 Bridge Error Status Register 1 (P3P43BI0_BESR1)

Figure 2-25. PLB3 to PLB4 Bridge Error Status Register 1 (P3P4BI0_BESR1)

0:7		Reserved	
8:9	SSET5	Secondary PLB Bus Slave Error Type for master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred (4x mode only) 10 Master 5 slave error occurred 11 Reserved	
10	R/W5	Read/write status master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read	
11	FLK5	P3P4BI0_BESR field lock master 5 0 Master 5 P3P4BI0_BESR field is unlocked 1 Master 5 P3P4BI0_BESR field is locked	
12	ALK5	P3P4BI0_BEAR address lock master 5 0 Master 5 P3P4BI0_BEAR address is unlocked 1 Master 5 P3P4BI0_BEAR address is locked	

13	WIRQ5	Write Error Interrupt master 5 0) No write error detected - master 5interrupt request is inactive 1) Write error detected - master 5interrupt request is active	
14:15	PAR5	Parity Error Type master 5 00) No parity errors detected for master 5 01) Read bus parity error detected from secondary PLB slave device read data for master 5 10) Address bus parity error or BE parity error detected from master 5 11) Write bus parity error detected from data supplied by master 5	
16:31		Reserved	

2.1.6.5 PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG)

Figure 2-26. PLB3 to PLB4 Bridge Configuration Register (P3P4BI0_CFG)

0	SSPS	Slave side primary/secondary priority 0 This PLB3-to-PLB4 Bridge Slave side is the secondary Slave side. 1 This PLB3-to-PLB4 Bridge Slave side is the primary Slave side.	
1:2	FIFOE	FIFO Enablement 00 PLB3-to-PLB4 Bridge Address FIFO is disabled. 01 PLB3-to-PLB4 Bridge Acknowledged Address FIFO is set to one level deep. 10 PLB3-to-PLB4 Bridge Acknowledged Address FIFO is set to two levels deep. 11 Reserved	These are read only bits
3	SRE	Secondary Read Enablement 0 0) PLB3-to-PLB4 Bridge secondary read feature is disabled. 1 1) PLB3-to-PLB4 Bridge secondary read feature is enabled.	Read only bit
4	SWE	Secondary Write Enablement 0 PLB3-to-PLB4 Bridge secondary write feature is disabled. 1 PLB3-to-PLB4 Bridge secondary write feature is enabled.	Read only bit
5	PBASS	PLB Bus Architecture for Slave Side 0 PLB3-to-PLB4 Bridge Slave side is connected to a 3.x PLB Bus. 1 PLB3-to-PLB4 Bridge Slave side is connected to a 4.x PLB Bus.	
6	PBAMS	PLB Bus Architecture for Master Side 0 PLB3-to-PLB4 Bridge Master side is connected to a 3.x PLB Bus. 1 PLB3-to-PLB4 Bridge Master side is connected to a 4.x PLB Bus.	

7	ER	Enable Rearbitration 0 PLB3-to-PLB4 Bridge Slave side rearbitration is disabled. 1 PLB3-to-PLB4 Bridge Slave side rearbitration is enabled.	Read only
8:9	PRI	PLB Priority Bits 00 Lowest 01 Next Highest 10 Next Highest 11 Highest	These bits determine the priority of PLB requests. They are directly connected to the PLB_priority(0:1) bits during a PLB request. During reset, these bits are set to the value present on the PGM_PPB3_priority input.
10	PE	Parity Enable 0 Parity checking is disabled 1 Parity checking is enabled on a per side basis depending on whether connected to a 4.x PLB Bus.	During reset this bit is set to the value present on PGM_PPB3_parEnable input.
11	FI	Flush Interrupt Enable 0 Interrupt flushing of writes is disabled and the interrupt will not be forwarded to B-PLB slave via FlushIntComp 1 Interrupt flushing of writes from A-PLB master X is enabled and will be forwarded to B-PLB slave via FlushIntComp	
12:14	FIID	Flush Interrupt ID	Value programmed must match the master ID that the A-PLB bus master X is connected to. (For example, if master X is connected to port 4 of the A-PLB arbiter then the FIID should be programmed to 3'b100)
15:31	Reserved		

2.1.6.6 PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR)

When PPBx_PICR[PICE] = 0, PLB3 to PLB4 master side request priority will not increment and is fixed at the programmed value, set in P3P4BI0_CFG[8:9].

When P3P4BI0_PICR[PICE] = 1, (this is the default after system reset) if the PLB3-to-PLB4 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. Then, if the PLB3-to-PLB4 master side active request is not acknowledged by a PLB slave device in the programmed time then the priority will increment to the next highest value. This continues until 2'b11 is reached. For next request, priority value starts out at the original value.

Initial Priority Incrementation Counter Value (1:5 bits): The PLB3-to-PLB4 bridge counts blocks of 8 PLB clocks for each value of the counter. Thus, for a count value of 0b00000 the counter would count 8 PLB clocks and increment its request priority if it has not received an acknowledgement from the target PLB slave device. A count value of 0b00001 would expire after 16 PLB clocks, a value of 0b00010 would expire after 24 PLB clocks, etc.

Figure 2-27. PLB3 to PLB4 Bridge Priority Incrementation Counter Register (P3P4BI0_PICR)

0	PICE	Priority Incrementation Counter Enable 0 Counter is disabled. 1 Counter is enabled.	
1:5	IPICV	Initial Priority Incrementation Counter Value The maximum count value, 5'b11111, is 256 PLB clocks, or 1.92 us at 133MHz.	Value resets to zero. The maximum count value, 0b11111, is 256 PLB clocks, or 1.92 us at 133MHz.
6:31		Reserved	

2.1.6.7 PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR)

Figure 2-28. PLB3 to PLB4 Bridge Parity Error Interrupt Register (P3P4BI0_PEIR)			
0	ADRPE	A Bus Parity Error Interrupt 0 No Upper address, lower address or byte enable parity errors detected. 1 Parity error detected on either the upper address, lower address of byte enables of requested transfer to PLB3-to-PLB4 Bridge. PLB3-to-PLB4 address parity error interrupt is active.	
1	RDPE	Read Data Bus Parity Error Interrupt 0 No read data bus parity error detected 1 Read data bus parity error detected from secondary PLB Bus slave device data transfer to PLB3-to-PLB4 Bridge. PLB3-to-PLB4 read data parity error interrupt is active.	
2:31		Reserved	

2.1.6.8 PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID)

Figure 2-29. PLB3 to PLB4 Bridge Revision ID Register (P3P4BI0_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

2.1.7 PLB4 to OPB Bridge Registers

The PLB4 to OPB bridge registers listed in the following figure are DCRs accessed using the **mfdcr** and **mtdcr** instructions. PLB4 to OPB0 bridge registers contain control and status for bus OPB1. PLB4 to OPB1 bridge registers contain control and status for bus OPB2.

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Table 2-10. PLB4 to OPB Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB42OPB0_BESR0	PLB4 to OPB0 Bridge Error Status Register 0 (Master IDs 0,1, 2, 3)	0x200	R/Clear	91
PLB42OPB0_BEARL	PLB4 to OPB0 Bridge Error Address Register Low	0x202	R/O	93
PLB42OPB0_BEARH	PLB4 to OPB0 Bridge Error Upper Address Register High	0x203	R/O	93
PLB42OPB0_BESR1	PLB4 to OPB0 Bridge Error Status Register (Master IDs 4, 5)	0x204	R/Clear	94
PLB42OPB0_CFG	PLB4 to OPB0 Bridge Configuration Register	0x206	R/Clear	95
PLB42OPB0_LATENCY	PLB4 to OPB0 Bridge Burst Latency Timer	0x208	R/Clear	95
PLB42OPB0_REVID	PLB4 to OPB0 Bridge Revision ID Register	0x20A	R/O	96
PLB42OPB1_BESR0	PLB4 to OPB1 Bridge Error Status Register 0 (Master IDs 0,1, 2, 3)	0x340	R/Clear	91
PLB42OPB1_BEARL	PLB4 to OPB1 Bridge Error Address Register Low	0x342	R/O	93
PLB42OPB1_BEARH	PLB4 to OPB1 Bridge Error Upper Address Register High	0x343	R/O	93
PLB42OPB1_BESR1	PLB4 to OPB1 Bridge Error Status Register (Master IDs 4, 5)	0x344	R/Clear	94
PLB42OPB1_CFG	PLB4 to OPB1 Bridge Error Configuration Register	0x346	R/Clear	95
PLB42OPB1_LATENCY	PLB4 to OPB1 Bridge Burst Latency Timer	0x348	R/Clear	95
PLB42OPB1_REVID	PLB4 to OPB1 Bridge Revision ID Register	0x34A	R/O	96

2.1.7.1 PLB4 to OPB Bridge Error Status Register 0 (PLB42OPBx_BESR0)

The PLB4 to OPB bridge writes error information into the appropriate PLB42OPBx_BESR*m* register. (For master IDs 0, 1, 2, and 3, *m* = 0; for master IDs 4 and 5, *m* = 1.)

PLB42OPBx_BESR*m* fields can be locked using the PLB42OPBx_BESR*m*[FLK*n*] and PLB42OPBx_BESR*m*[ALK*n*] fields (*n* is the master ID). Once locked, the PLB42OPBx_BESR*m* fields associated with a master cannot be overwritten if a subsequent error occurs until the locking fields are cleared. To clear a lock, write 1 to the PLB42OPBx_BESR*m*[FLK*n*] and PLB42OPBx_BESR*m*[ALK*n*] fields that are set. Writing 0 to a lock field does not affect the field.

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Figure 2-30. PLB4 to OPB Bridge Error Status Register 0 (PLB42OPBx_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	PLB4 master 0 is the read-only instruction cache unit (ICU)
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	

3	FLK0	PLB42OPBx_BESR0 Field Lock Master 0 0 Master 0 PLB42OPBx_BESR0 field is unlocked 1 Master 0 PLB42OPBx_BESR0 field is locked	
4	ALK0	PLB42OPBx_BEAR Address Lock Master 0 0 Master 0 PLB42OPBx_BEAR address is unlocked 1 Master 0 PLB42OPBx_BEAR address is locked	
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active	
6:7		Reserved	
8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved	PLB4 master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read	
11	FLK1	PLB42OPBx_BESR0 Field Lock Master 1 0 Master 1 PLB42OPBx_BESR0 field is unlocked 1 Master 1 PLB42OPBx_BESR0 field is locked	
12	ALK1	PLB42OPBx_BEAR Address Lock Master 1 0 Master 1 PLB42OPBx_BEAR address is unlocked 1 Master 1 PLB42OPBx_BEAR address is locked	
13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active	
14:15		Reserved	
16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	PLB4 master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	PLB42OPBx_BESR0 Field Lock Master 2 0 Master 2 PLB42OPBx_BESR0 field is unlocked 1 Master 2 PLB42OPBx_BESR0 field is locked	
20	ALK2	PLB42OPBx_BEAR Address Lock Master 2 0 Master 2 PLB42OPBx_BEAR address is unlocked 1 Master 2 PLB42OPBx_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	

24:25	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	PLB4 master 3 is the DMA2P40 unit
26	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
27	FLK3	PLB42OPBx_BESR0 Field Lock Master 3 0 Master 3 PLB42OPBx_BESR0 field is unlocked 1 Master 3 PLB42OPBx_BESR0 field is locked	
28	ALK3	PLB42OPBx_BEAR Address Lock Master 3 0 Master 3 PLB42OPBx_BEAR address is unlocked 1 Master 3 PLB42OPBx_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active	
30:31		Reserved	

2.1.7.2 PLB4 to OPB Bridge Error Address Register Low (PLB42OPBx_BEARL)

The read-only PLB42OPBx_BEARL reports the lower 32 bits of the address of a PLB4 to OPB transfer that results in an error. The PLB4 to OPB bridge writes the error address in the PLB42OPBx_BEARL, unless the associated PLB42OPBx_BESR*m*[ALCK*n*] field is set (*m* is either 0 or 1, depending on the master ID specified by *n*). Once locked, the PLB4 to OPB bridge cannot write PLB42OPBx_BEARL until all PLB42OPBx_BESR*m*[ALCK*n*] fields that are set are cleared.

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Figure 2-31. PLB4 to OPB Bridge Error Address Register Low (PLB42OPBx_BEARL)

0:31	BEARL	Lower address of bus error	

2.1.7.3 PLB4 to OPB Bridge Error Address Register High (PLB42OPBx_BEARH)

The read-only PLB42OPBx_BEARH reports the upper four address bits of a PLB4 to OPB transfer that results in an error. The PLB4 to OPB bridge writes the error address in the PLB42OPBx_BEARH, unless the associated PLB42OPBx_BESR*m*[ALCK*n*] field is set (*m* is either 0 or 1, depending on the master ID specified by *n*). Once locked, the PLB4 to OPB bridge cannot write PLB42OPBx_BEARH until all PLB42OPBx_BESR*m*[ALCK*n*] fields that are set are cleared.

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Figure 2-32. PLB4 to OPB Bridge Error Address Register High (PLB42OPBx_BEARH)

0:27		Reserved	
28:31	UA	Upper address of bus error	

2.1.7.4 PLB4 to OPB Bridge Error Status Register 1(PLB42OPBx_BESR1)

Figure 2-33. PLB4 to OPB Bridge Error Status Register 1 (PLB42OPBx_BESR1)

0:1	PTE4	PLB Timeout Error Status Master 4 00 No error occurred 01 Timeout error occurred 10 Slave error occurred 11 Reserved	Master 4 is the PLB3 to PLB4 bridge.
2	R/W4	Read/Write Status Master 4 0 Error operation is a write 1 Error operation is a read	
3	FLK4	PLB42OPBx_BESR1 Field Lock Master 4 0 PLB42OPBx_BESR1 field is unlocked 1 PLB42OPBx_BESR1 field is locked	
4	ALK4	PLB42OPBx_BEAR Address Lock Master 4 0 PLB42OPBx_BEAR address is unlocked 1 PLB42OPBx_BEAR address is locked	
5	WIRQ4	Write Error Interrupt Master 4 0 No write error detected - interrupt request is inactive 1 Write error detected - interrupt request is active	
6:7		Reserved	
8:9	PTE5	PLB Timeout Error Status Master 5 00 No error occurred 01 Timeout error occurred 10 Slave error occurred 11 Reserved	Master 5 is OPB to PLB bridge 1.
10	R/W5	Read/Write Status Master 5 0 Error operation is a write 1 Error operation is a read	
11	FLK5	PLB42OPBx_BESR1 Field Lock Master 5 0 PLB42OPBx_BESR1 field is unlocked 1 PLB42OPBx_BESR1 field is locked	
12	ALK5	PLB42OPBx_BEAR Address Lock Master 5 0 PLB42OPBx_BEAR address is unlocked 1 PLB42OPBx_BEAR address is locked	
13	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - interrupt request is inactive 1 Write error detected - interrupt request is active	
14:15		Reserved	
16:17	PTE6	PLB Timeout Error Status Master 6 00 No error occurred 01 Timeout error occurred 10 Slave error occurred 11 Reserved	Master 6 is MAL.
18	R/W6	Read/Write Status Master 6 0 Error operation is a write 1 Error operation is a read	
19	FLK6	PLB42OPBx_BESR1 Field Lock Master 6 0 PLB42OPBx_BESR1 field is unlocked 1 PLB42OPBx_BESR1 field is locked	

20	ALK6	PLB42OPBx_BEAR Address Lock Master 6 0 PLB42OPBx_BEAR address is unlocked 1 PLB42OPBx_BEAR address is locked	
21	WIRQ6	Write Error Interrupt Master 6 0 No write error detected - interrupt request is inactive 1 Write error detected - interrupt request is active	
22:23		Reserved	
24:25	PTE7	PLB Timeout Error Status Master 7 00 No error occurred 01 Timeout error occurred 10 Slave error occurred 11 Reserved	Master 7 is Security Engine.
26	R/W7	Read/Write Status Master 7 0 Error operation is a write 1 Error operation is a read	
27	FLK7	PLB42OPBx_BESR1 Field Lock Master 7 0 PLB42OPBx_BESR1 field is unlocked 1 PLB42OPBx_BESR1 field is locked	
28	ALK7	PLB42OPBx_BEAR Address Lock Master 7 0 PLB42OPBx_BEAR address is unlocked 1 PLB42OPBx_BEAR address is locked	
29	WIRQ7	Write Error Interrupt Master 7 0 No write error detected - interrupt request is inactive 1 Write error detected - interrupt request is active	
30:31		Reserved	

2.1.7.5 PLB4 to OPB Bridge Configuration Register (PLB42OPBx_CFG)

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Figure 2-34. PLB4 to OPB Bridge Configuration Register (PLB42OPBx_CFG)

0	ER	Enable rearbitration 0 PLB4 to OPB bridge rearbitration is enabled 1 PLB4 to OPB bridge rearbitration is disabled
1	L32R	Line 32 bit read 0 PLB4 to OPB bridge reads line operations at requested size of master 1 PLB4 to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

2.1.7.6 PLB4 to OPB Bridge Burst Latency Timer (PLB42OPBx_LATENCY)

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Figure 2-35. PLB4 to OPB Bridge Burst Latency Timer Register (PLB42OPBx_LATENCY)			
0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled	
1:4	LC	Latency Count 0000 - Minimum count value. 1111 - Maximum count value. PLB4 to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.	When PLB42OPBx_LATENCY[LC=0000] PLB4 to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. When PLB42OPBx_LATENCY[LC=1111] PLB4 to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved	

2.1.7.7 PLB4 to OPB Bridge Revision ID (PLB42OPBx_REVID)

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

Figure 2-36. PLB4 to OPB Bridge Revision ID Register (PLB42OPBx_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL (hard wired to 0x1)
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL (hard wired to 0x22)

2.1.8 PLB3 Arbiter Registers

PLB3 arbiter registers are DCRs accessed using the **mfdcr** and **mtdcr** instructions.

Table 2-11. PLB3 Arbiter 0 Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB3A0_REVID	PLB3 Arbiter 0 Revision ID Register	0x072	R	96
PLB3A0_BESR	PLB3 Error Status Register	0x074	R/W	97
PLB3A0_BEAR	PLB3 Error Address Register	0x076	R	98
PLB3A0_ACR	PLB3 Arbiter Control Register	0x077	R/W	98

2.1.8.1 PLB3 Arbiter Revision ID Register (PLB3A0_REVID)

PLB3A0_REVID is a 32-bit read-only register that contains the revision ID of the PLB3 arbiter. The contents of the register can be accessed by using the move from device control register (mfdcr) instruction. The PLB3A0_REVID register can be accessed with CPU_dcrAddr(6:9) = 0x2. The register is not affected by reset.

Figure 2-37. PLB3 Arbiter 0 Revision ID Register (PLB3A0_REVID)			
0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

2.1.8.2 PLB3 Error Status Register (PLB3A0_BESR)

The read/clear PLB3A0_BESR identifies timeout errors on PLB3 bus transfers, the master initiating the transfer, and the type of transfer.

The PCI Bridge master can qualify its PLB transactions such that the information describing any timeout errors that occur during these transfers becomes locked. Once locked, a master's PLB3A0_BESR[PTEn] and PLB3A0_BESR[R/Wn] fields cannot be updated until the corresponding PLB3A0_BESR[FLKn] field is cleared. To clear a PLB3A0_BESR field, write 1 to the field. Writing 0 to a PLB3A0_BESR field does not affect the field.

For PCI bridge timeout errors to become locked, an error must occur while the PCI bridge has error locking enabled (PCIC0_BRDGOPT1[PLESE] = 1).

Figure 2-38. PLB3 Error Status Register (PLB3A0_BESR)			
0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	PLB3 master 0 is PLB4 to PLB3 bridge.
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB3A0_BESR Field Lock 0 Master 0 PLB3A0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB3A0_BEAR Address Lock 0 Master 0 PLB3A0_BEAR is unlocked 1 Master 0 PLB3A0_BEAR is locked	
4:7		Reserved	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is PCI bridge.
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB3A0_BESR Field Lock 0 Master 2 PLB3A0_BESR field is unlocked 1 Master 2 PLB3A0_BESR field is locked	
11	ALK2	Master 2 PLB3A0_BEAR Address Lock 0 Master 2 PLB3A0_BEAR is unlocked 1 Master 2 PLB3A0_BEAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	PLB3 master 3 is DMA.

13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB3A0_BESR Field Lock 0 Master 3 PLB3A0_BESR field is unlocked 1 Master 3 PLB3A0_BESR field is locked	
15	ALK3	Master 3 PLB3A0_BEAR Address Lock 0 Master 3 PLB3A0_BEAR is unlocked 1 Master 3 PLB3A0_BEAR is locked	
16:19		Reserved	
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	PLB3 master 5 is EBC.
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB3A0_BESR Field Lock 0 Master 5 PLB3A0_BESR field is unlocked 1 Master 5 PLB3A0_BESR field is locked	
23	ALK5	Master 5 PLB3A0_BEAR Address Lock 0 Master 5 PLB3A0_BEAR is unlocked 1 Master 5 PLB3A0_BEAR is locked	
24:31		Reserved	

2.1.8.3 PLB3 Error Address Register (PLB3A0_BEAR)

The read-only PLB3A0_BEAR contains the address of the access on which a bus timeout error occurred. The PCI bridge master can qualify its PLB transactions so that a bus timeout error causes PLB3A0_BEAR to become locked. Once locked, the PLB3A0_BEAR cannot be updated if a subsequent error occurs, until all PLB3A0_BESR[FLCK*n*] fields are cleared (*n* is the master ID).

Figure 2-39. PLB3 Error Address Register (PLB3A0_BEAR)

0:31	BEAR	Address of bus timeout error	

2.1.8.4 PLB3 Arbiter 0 Control Register (PLB3A0_ACR)

PLB3A0_ACR controls PLB arbitration priority, which is determined by PLB priority mode and PLB priority order.

Figure 2-40. PLB3 Arbiter Control Register (PLB3A0_ACR)

0	PPM	PLB Priority Mode 0 Fixed 1 Fair	

1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 4, 5 001 Masters 1, 2, 3, 4, 5, 0 010 Masters 2, 3, 4, 5, 0, 1 011 Masters 3, 4, 5, 0, 1, 2 100 Masters 4, 5, 0, 1, 2, 3 101 Masters 5, 0, 1, 2, 3, 4 110 Reserved 111 Reserved	
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	
5:31		Reserved	

The PLB3 arbiter supports fixed and fair priority schemes to handle situations when two or more PLB3 masters simultaneously present requests of the same priority. In both fixed and fair modes, the arbiter services pending requests in the order specified by PLB3A0_ACR[PPO]. Fair mode differs in that the arbiter automatically updates PLB3A0_ACR[PPO] after arbitration cycles where multiple masters requested at the same priority. The value written into PLB3A0_ACR[PPO] gives the next to highest priority-requesting master the highest priority for the next arbitration cycle. For example, if the arbiter is operating in fair mode, PLB3A0_ACR[PPO]=0b011, and masters 0, 3, and 4 request service at the same priority level, master 3 is granted the PLB. Subsequently, PLB3A0_ACR[PPO] is set to 0b100, giving master 4 the highest priority for the next arbitration cycle.

Enabling PLB3A0_ACR[HBU] maximizes the total PLB data bus throughput when there are active master requests for the PLB3 read and write data buses pending with one data bus busy and the other idle. If a primary transfer is busy using one of the data buses, a secondary request for that bus has been acknowledged, and a third request destined for the same bus is the next highest priority request a lower priority master requesting use of the other bus would be "blocked" from gaining use of the address and transfer qualifiers bus. When PLB3A0_ACR[HBU]=1 the PLB3 arbiter internally promotes the lower priority master to the level of the currently active master thus allowing the lower priority master to utilize the idle bus.

2.1.9 PLB3 to OPB Bridge Registers

The PLB3 to OPB bridge registers listed in the following table are DCRs accessed using the mfdcr and mtdcr instructions. PLB3 to OPB bridge and OPB to PLB3 bridges are identical.

Table 2-12. PLB3 to OPB Bridge Registers

Mnemonic	Register Name	DCR Address	Access	Page
PLB32OPB0_BESR0	PLB3 to OPB Bridge Error Status Register 0 (Master IDs 0, 1, 2, 3)	0x0090	R/Clear	100
PLB32OPB0_BEAR	PLB3 to OPB Bridge Error Address Register	0x0092	R	101
PLB32OPB0_REVID	PLB3 to OPB Revision ID Register	0x0093	R	101
PLB32OPB0_BESR1	PLB3 to OPB Bridge Error Status Register 1 (Master IDs 4, 5)	0x0094	R/Clear	102

PLB32OPB0_BEAR and thePLB32OPB0_BESRn registers record information on errors that occur during PLB3 master transactions that target OPB address space. When an error is detected, the address is saved in the PLB32OPB0_BEAR, and details on the type of error are logged in one of the PLB32OPB0_BESRn registers.

Unlike the other PLB3 masters, the PCI bridge can qualify PLB3 transactions with a lock request, so that slave error status information becomes locked. The PCI bridge requests error locking for its PLB3 transfers when PCIC0_BRDGOPT1[PLESE] = 1.

When an error occurs during a PLB3 transaction with error locking enabled and PLB32OPB0_BEAR is not already locked, the address of the error is stored in PLB32OPB0_BEAR and the address lock bit (PLB32OPB0_BESR*m*[ALK*n*]) is set for the master whose transaction caused the error. (*m* is either 0 or 1, depending on the master ID specified by *n*) At the same time, if the field lock bit is not set (PLB32OPB0_BESR*m*[FLK*n*]=0), the type of error is saved in PLB32OPB0_BESR*m*[PTE*n*, R/W*n*], and these fields can then be locked by setting PLB32OPB0_BESR*m*[FLK*n*] = 1. When software processes the error, it should clear the error status and both lock bits at the same time.

2.1.9.1 PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0)

Errors detected by the PLB3 to OPB bridge are recorded in the appropriate PLB32OPB0_BESRx register. PLB32OPB0_BESR0 records errors for master ID 0, 1, 2, and 3.

When PLB32OPB0_BESR0[FLK*n*] is set, subsequent errors cannot update PLB32OPB0_BESR0[PTE*n*, R/W*n*]. Similarly, if any PLB32OPB0_BESR0[ALK*n*] = 1, PLB32OPB0_BEAR is locked. Once locked, the PLB32OPB0_BESR0 fields associated with a master cannot be overwritten if a subsequent error occurs until the locking fields are cleared. To clear a lock, write 1 to the PLB32OPB0_BESR0[FLK*n*] and PLB32OPB0_BESR0[ALK*n*] fields that are set. Writing 0 to a lock field does not affect the field.

For PLB32OPB0_BESR0[FLK*n*, AL*n*] to become locked, an error must occur while the PCI bridge has error locking enabled (PCIC0_BRDGOPT1[PLESE] = 1).

Figure 2-41. PLB3 to OPB Bridge Error Status Register 0 (PLB32OPB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved	Master 0 - PLB4 to PLB3
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read	
3:9		Reserved	
10:11	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 - PCI Bridge
12	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
13	FLK2	PLB32OPB0_BESR0 Field Lock Master 2 0 Master 2 PLB32OPB0_BESR0 field is unlocked 1 Master 2 PLB32OPB0_BESR0 field is locked	
14	ALK2	PLB32OPB0_BEAR Address Lock Master 2 0 Master 2 PLB32OPB0_BEAR address is unlocked 1 Master 2 PLB32OPB0_BEAR address is locked	
15:16	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	Master 3 - DMA2P30

17	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
18:31		Reserved	

2.1.9.2 PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR)

The read-only PLB32OPB0_BEAR reports the address of a PLB3 to OPB transfer that results in an error. Errors detected by the PLB to OPB bridge are recorded in the PLB32OPB0_BEAR, unless the associated PLB32OPB0_BESR*m*[ALK*n*] field was previously set. (*m* is either 0 or 1, depending on the master ID specified by *n*. For master IDs 0, 1, 2, and 3, *m* = 0; for master IDs 4 and 5 *m* = 1.) An address lock field is set when a PLB master requests error locking, and the PLB3 to OPB Bridge detects an error during transaction. The only master that can request error locking is the PCI Bridge. Once locked, the PLB3 to OPB bridge cannot write PLB32OPB0_BEAR until all PLB32OPB0_BESR*m*[ALK*n*] fields that are set are cleared.

Figure 2-42. PLB3 to OPB Bridge Error Address Register (PLB32OPB0_BEAR)

0:31	BEAR	Address of bus error	

2.1.9.3 PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID)

The PLB32OPB0_REVID register described in the following figure is a 32-bit read-only register that contains the revision ID of the PLB3 to OPB Bridge. The contents of the register can be accessed by using the move from device control register (mfdcr) instruction. The register is not affected by reset.

Figure 2-43. PLB3 to OPB Bridge Revision ID Register (PLB32OPB0_REVID)

0:27		Reserved	
28:31	RID	This value is the revision level for PLB32OPB0.	

2.1.9.4 PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1)

Errors detected by the PLB3 to OPB bridge are recorded in the appropriate PLB32OPB0_BESRx register. PLB32OPB0_BESR1 records errors for master IDs 4 and 3.

Figure 2-44. PLB3 to OPB Bridge Error Status Register 1 (PLB32OPB0_BESR1)

0:2		Reserved	
3:4		Reserved	
5:6	PTE5	PLB3 Timeout Error Status Master 5 00 No Master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved	Master 5 - External Bus Master Interface (EBMI)
7	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read	
8:31		Reserved	

2.2 On-Chip Peripheral Bus

The OPB attaches peripherals that do not have the high bandwidth or low latency requirements that would justify their direct attachment to the PLB. OPB segments interface to the PLB via the PLB to OPB bridges. These bridges allow PLB masters to access OPB slaves. Reverse bridges (OPB to PLB) allow OPB masters to access PLB slaves. PPC440EPx implements three OPB segments, two attached to the PLB4 and the one attached to the PLB3. The PPC440EPx implements one OPB segment attached to the PLB3.

Note: PLB4 to OPB Bridge is supported by the PPC440EPx Only.

2.2.1 OPB Features

The on-chip peripheral bus features:

- A 32-bit address bus and a 32-bit data bus
- Dynamic bus sizing; byte, halfword, and fullword transfers
- Byte and halfword duplication for byte and halfword transfers
- Single-cycle transfer of data between OPB bus master and OPB slaves
- Sequential address (burst) protocol support
- A 16-cycle fixed bus time-out provided by the OPB arbiter
- Bus parking for reduced latency
- Bus arbitration overlapped with last cycle of bus transfers

2.2.2 OPB Master Assignments

Table 2-13 lists the three masters supported by OPBA0.

Table 2-13. Master Assignments

OPB Agents	Description
OPBA0, OPB Arbiter for OPB0	
PLB to OPB Bridge Controller	PLB32OPB0 (master 0)
Direct Memory Access Controller	DMA2P30 (master 1)
OPBA1, OPB Arbiter for OPB1 *(PPC440EPx only)*	
Direct Memory Access Controller	DMA2P40 (master 0)
PLB4 to OPB Bridge controller 0	PLB42OPB0 (master 1)
OPBA2, OPB Arbiter for OPB2 *(PPC440EPx only)*	
USB 2.0 Host	USB2H0 (master 0)
PLB4 to OPB Bridge Controller 1	PLB42OPB1 (master 1)

2.2.3 OPB Arbiter Registers

PPC440EPx implements three OPB arbiters: OPBA0 is attached to PLB3, and OPBA1 and OPBA2 are attached to PLB4. PPC440GRx implements one OPB arbiter: OPBA0 is attached to PLB3. The addresses for these arbiters' priority and control registers are provided in the following table.

Table 2-14. OPB Arbiter Registers

Mnemonic	Register Name	Address	Access	Page
OPBA0_PR	OPB 0 Arbiter Priority Register	0x1 EF60 0A00	R/W	103
OPBA0_CR	OPB 0 Arbiter Control Register	0x1 EF60 0A01	R/W	104
OPBA1_PR	OPB 1 Arbiter Priority Register	0x0 E000 0000	R/W	103
OPBA1_CR	OPB 1 Arbiter Control Register	0x0 E000 0001	R/W	104
OPBA2_PR	OPB 2 Arbiter Priority Register	0x0 E000 0200	R/W	103
OPBA2_CR	OPB 2 Arbiter Control Register	0x0 E000 0201	R/W	104

Other than for their address space, the arbiter registers are identical. Hence, in the following topics, they share common definitions as registers OPBAx_PR and OPBAx_CR.

2.2.3.1 OPB Arbiter Priority Register (OPBAx_PR)

The OPBAx_PR assigns priorities to the OPB master IDs.

Note: OPBA1 and OPBA2 are supported by the PPC440EPx Only.

Figure 2-45. OPB Arbiter Priority Register (OPBAx_PR)			
0:1	HPM	High Priority Master ID 00 Master ID 0of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved	For the OPBA0 arbiter, this priority is assigned to the PLB3-to-OPB bridge at reset. For the OPBA1 arbiter, this priority is assigned to the DMA2PLB4 controller at reset. For the OPBA2 arbiter, this priority is assigned to the USB 2.0 Host at reset.
2:3	MHP	Medium High Priority master ID 00 Master ID of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID of OPB master device connected to M1_request and OPB_M1Grant 10 Master ID of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3Grant	For the OPBA0 arbiter, this priority is assigned to the DMA2PLB3 at reset. For the OPBA1 arbiter, this field priority is assigned to the PLB4-to-OPB Bridge Controller 0 at reset. For the OPBA2 arbiter, this priority is assigned to PLB4-to-OPB Bridge Controller 1 at reset.
4:7		Reserved	

2.2.3.2 OPB Arbiter Control Register (OPBAx_CR)

The OPBAx_CR fields controls updating of the OPBAx_PR (described in *OPB Arbiter Priority Register (OPBAx_PR)* on page 103).

Figure 2-46. OPB Arbiter Control Register (OPBAx_CR)			
0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled	
1	PEN	Park Enable 0 Park disabled 1 Park enabled	
2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last	
3:4	PID	Parked Master ID 00 Master ID 0 of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID 1 of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2 of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID 3 of OPB master device connected to M3_request and OPB_M3GrantReserved	
5:7		Reserved	

2.2.4 OPB-to-PLB4 Bridge Registers

Note: OPB to PLB4 Bridge is supported by the PPC440EPx Only.

Table 2-15. OPB-to-PLB4 Bridge Registers

Mnemonic	Register Name	Address	Access	Page
OPB2PLB40_BCTRL	OPB-to-PLB4 Bridge Control Register	0x0350	R/W	105
OPB2PLB40_BSTAT	OPB-to-PLB4 Bridge Status Register	0x0351	R	105
OPB2PLB40_BEARL	OPB-to-PLB4 Error Address Register Low	0x0352	R	106
OPB2PLB40_BEARH	OPB-to-PLB4 Error Address Register High	0x0353	R	106
OPB2PLB40_REVID	OPB-to-PLB4 Bridge Revision ID Register	0x0354	R	106

2.2.4.1 OPB-to-PLB4 Bridge Control Register (OPB2PLB40_BCTRL)

The OPB-to-PLB4 bridge control register contains the control bits for the OPB-to-PLB4 bridge. It is read and written by means of the DCR bus.

Note: OPB to PLB4 Bridge is supported by the PPC440EPx Only.

Figure 2-47. OPB-to-PLB4 Bridge Control Register (OPB2PLB40_BCTRL)

0:1	PRI	PLB Priority Bits. 00 Lowest 01 10 11 Highest	Used to determine the priority of OPB to PLB bridge requests on the PLB
2:31		Reserved	

2.2.4.2 OPB-to-PLB4 Bridge Status Register (OPB2PLB40_BSTAT)

The OPB-to-PLB4 bridge status register contains the error bit for the OPB-to-PLB4 bridge. To clear the status register, a 1 must be loaded into those register bits that are to be cleared. Writing a 0 to any bit in the status register does not affect the state of the bit.

Note: OPB to PLB4 Bridge is supported by the PPC440EPx Only.

Figure 2-48. OPB-to-PLB4 Bridge Status Register (OPB2PLB40_BSTAT)			
0	RDERR	Read Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a read error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB-to-PLB4 bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1	WRERR	Write Error Bit Reset to 0	This bit is set whenever a PLB slave device reports a write error to the bridge. For posted single writes and errors occurring after the deassertion of OPB_SeqAddr on burst writes, no error is reported to the OPB master through BGI_ErrAck.An error is reported to the OPB master through BGI_ErrAck for a previous write error if OPB_SeqAddr is still asserted (note that the error does not correspond to the transfer during which it is reported).
2:31		Reserved	

2.2.4.3 OPB-to-PLB4 Bridge Error Address Register Low (OPB2PLB40_BEARL)

Note: OPB to PLB4 Bridge is supported by the PPC440EPx Only.

Figure 2-49. OPB-to-PLB4 Bridge Error Address Register Low (OPB2PLB40_BEARL)			
0:31		Bridge Error Address Register Low	This is a read only register. As part of its error reporting, the OPB-to-PLB4 bridge loads the OPB2PLB40_BEARL with the error address if the ACLK (address lock bit) is not already set.

2.2.4.4 OPB-to-PLB4 Bridge Error Address Register High (OPB2PLB40_BEARH)

Note: OPB to PLB4 Bridge is supported by the PPC440EPx Only.

Figure 2-50. OPB-to-PLB4 Bridge Error Address Register High (OPB2PLB40_BEARH)			
0:27		Reserved	
28:31	ERRADD	Bridge Error Address Register High	This is a read only register. As part of its error reporting, the OPB-to-PLB4 bridge loads the OPB2PLB40_BEARH with the error address if the ACLK (address lock bit) is not already set.

2.2.4.5 OPB-to-PLB4 Bridge Revision ID Register (OPB2PLB40_REVID)

The OPB-to-PLB4 bridge revision ID register is a read-only register.

Note: OPB to PLB4 Bridge is supported by the PPC440EPx Only.

Figure 2-51. OPB-to-PLB4 Bridge Revision ID Register (OPB2PLB40_REVID)			
0:27		Reserved	
28:31	RID	This value indicates the revision level of the OPB-to-PLB4 bridge.	

2.3 Device Control Register (DCR) Bus

The DCR bus provides a mechanism for the PPC440EPx/GRx Embedded Processor to setup other on-chip facilities. For example, programmable resources in the DDR-SDRAM controller are configured for use with various memory devices according to their transfer characteristics and address assignments. DCR's are accessed through the use of the PowerPC mfdcr and mtdcr instructions.

The DCR bus also allows the PPC440 CPU to communicate with peripheral devices without using the PLB interface, thereby avoiding the impact to the primary system bus bandwidth, and without additional segmentation of the useable address map.

Part II. PPC440EPx/GRx RISC Processor

3. Programming Model

The programming model of the PPC440EPx/GRx chip corresponds to the programming model of the PPC440 CPU core. The *PPC440 Processor User's Manual* describes how the following features and operations of the processor core appear to programmers:

- Storage addressing (including data types and byte ordering)
- Registers
- Instruction classes
- Instruction set
- Branch processing
- Integer processing
- Processor control
- User and supervisor state
- Speculative access
- Synchronization

3.1 Storage Addressing

As a 32-bit implementation of the Book-E Enhanced PowerPC Architecture, the PPC440EPx/GRx Embedded Processor implements a uniform 32-bit effective address (EA) space. Effective addresses are expanded into virtual addresses and then translated to 33-bit (8GB) real addresses by the memory management unit.

The PPC440EPx/GRx generates an effective address whenever it executes a storage access, branch, cache management, or translation look aside buffer (TLB) management instruction, or when it fetches the next sequential instruction.

3.2 User and Privileged Programming Models

The PPC440 Processor executes programs in two modes, also referred to as states. Programs running in privileged mode (also referred to as the supervisor state) can access any register and execute any instruction. These instructions and registers comprise the privileged programming model. In user mode, certain registers and instructions are unavailable to programs. This is also called the problem state. Those registers and instructions that are available comprise the user programming model. Privileged mode provides operating system software access to all processor resources. Because access to certain processor resources is denied in user mode, application software runs in user mode. Operating system software and other application software is protected from the effects of an errant application program.

Access to Device Control Registers (DCRs) described throughout this user's manual require the PPC440 Processor to be in supervisor state. Supervisor and problem state are described in more detail in the PPC440 Processor User's Manual.

Table 3-1. System Memory Address Map (Sheet 1 of 2)

Function	Sub Function	Start Address	End Address	Size
Total System Memory Address Space		0 0000 0000	1 FFFF FFFF	8GB
Local Memory	DDR SDRAM	0 0000 0000	0 7FFF FFFF	2GB
	Reserved	0 8000 0000	0 DFFF FFFF	
USB 2.0 Device *(PPC440EPx only)*	OPB1 Arbiter for USB	0 E000 0000	0 E000 003F	64B
	Reserved	0 E000 0040	0 E000 00FF	
	Device Controller	0 E000 0100	0 E000 017F	128B
	Reserved	0 5000 0180	0 5000 01FF	
USB 2.0 Host *(PPC440EPx only)*	OPB2 Arbiter for USB	0 E000 0200	0 E000 023F	64B
	Reserved	0 E000 0240	0 E000 02FF	
	Host EHIC	0 E000 0300	0 E000 03FF	256B
	Host OHIC	0 E000 0400	0 E000 04FF	256B
	Reserved	0 E000 0500	0 E000 FFFF	
On-Chip Memory	SRAM	0 E001 0000	0 E001 3FFF	16KB
	Reserved	0 E001 4000	0 E00F FFFF	
Security (PPC440EPx-S and PPC440GRx-S parts only)	Security Function	0 E010 0000	0 E017 FFFF	512KB
	Kasumi Algorithm	0 E018 0000	0 E018 07FF	2KB
	Reserved	0 E018 0800	1 7FFF FFFF	
PCI [1]	Memory	1 8000 0000	1 BFFF FFFF	1GB
EBC [1]	Controller	1 C000 0000	1 DFFF FFFF	512MB
	Reserved	1 E000 0000	1 E7FF FFFF	
PCI [1]	I/O	1 E800 0000	1 E800 FFFF	64KB
	Reserved	1 E801 0000	1E87F FFFF	
	I/O	1 E880 0000	1 EBFF FFFF	56MB
	Reserved	1 EC00 0000	1 EEBF FFFF	
	Configuration Registers	1 EEC0 0000	1 EEC0 0007	8B
	Reserved	1 EEC0 0008	1 EECF FFFF	
	Interrupt Ack / Special Cycle	1 EED0 0000	1 EED0 0003	4B
	Reserved	1 EED0 0004	1 EF3F FFFF	
	Local Configuration Registers	1 EF40 0000	1 EF40 003F	64B
	Reserved	1 EF40 0040	1 EF4F FFFF	

Table 3-1. System Memory Address Map (Sheet 2 of 2)

Function	Sub Function	Start Address	End Address	Size
	Reserved	1 EF50 0000	1 EF5F FFFF	
	General Purpose Timer	1 EF60 0000	1 EF60 01FF	512B
	Reserved	1 EF60 0200	1 EF60 02FF	
	UART0	1 EF60 0300	1 EF60 0307	8B
	Reserved	1 EF60 0308	1 EF60 03FF	
	UART1	1 EF60 0400	1 EF60 0407	8B
	Reserved	1 EF60 0408	1 EF60 04FF	
	UART2	1 EF60 0500	1 EF60 0507	8B
	Reserved	1 EF60 0508	1 EF60 05FF	
	UART3	1 EF60 0600	1 EF60 0607	8B
	Reserved	1 EF60 0608	1 EF60 06FF	
	IIC0	1 EF60 0700	1 EF60 071F	32B
	Reserved	1 EF60 0720	1 EF60 07FF	
	IIC1	1 EF60 0800	1 EF60 081F	32B
	Reserved	1 EF60 0820	1 EF60 08FF	
Internal Peripherals	SPI	1 EF60 0900	1 EF60 0906	6B
	Reserved	1 EF60 0907	1 EF60 09FF	
	OPB0 Arbiter	1 EF60 0A00	1 EF60 0A3F	64B
	Reserved	1 EF60 0A40	1 EF60 0AFF	
	GPIO0 Controller	1 EF60 0B00	1 EF60 0B7F	128B
	Reserved	1 EF60 0B80	1 EF60 0BFF	
	GPIO1 Controller	1 EF60 0C00	1 EF60 0C7F	128B
	Reserved	1 EF60 0C80	1 EF60 0CFF	
	Ethernet PHY ZMII	1 EF60 0D00	1 EF60 0D0F	16B
	Reserved	1 EF60 0D10	1 EF60 0DFF	
	Ethernet 0 Controller	1 EF60 0E00	1 EF60 0E77	120B
	Reserved	1 EF60 0E78	1 EF60 0EFF	
	Ethernet 1 Controller	1 EF60 0F00	1 EF60 0F77	120B
	Reserved	1 EF60 0F78	1 EF60 0FFF	
	Ethernet PHY RGMII	1 EF60 1000	1 EF60 1103	264B
	Reserved	1 EF60 1080	1 EFFF FFFF	
EBC [1]		1 F000 0000	1 FFDF FFFF	254MB
Boot space	EBC Bank 0 or PCI	1 FFE0 0000	1 FFFF FFFF	2MB

Notes:
3. EBC and PCI are relocatable, but this map reflects the suggested configuration.

Table 3-2. DCR Address Map (4KB of Device Configuration Registers)

Function	Start Address	End Address	Size
Total DCR Address Space[1]	000	3FF	1KW (4KB)[1]
By function:			
Reserved	000	00B	12W
Clocking Power On Reset (CPR0)	00C	00D	2W
System DCRs (SDR0)	00E	00F	2W
Memory Controller (SDRAM0)	010	011	2W
External Bus Controller (EBC0)	012	013	2W
Reserved	014	01F	12W
PLB4-to-PLB3 Bridge	020	02F	16W
PLB3-to-PLB4 Bridge	030	03F	16W
Reserved	040	06F	48W
PLB3 Arbiter	070	07F	16W
PLB4 Arbiter	080	08F	16W
PLB3-to-OPB0 Bridge	090	09F	16W
Reserved	0A0	0AF	16W
Power Management	0B0	0B7	8W
Reserved	0B8	0BF	8W
Interrupt Controller 0	0C0	0CF	16W
Interrupt Controller 1	0D0	0DF	16W
Interrupt Controller 2	0E0	0EF	16W
Power Management 1	0F0	0F7	8W
Reserved	0F8	0FF	8W
DMA-to-PLB3 Controller	100	13F	64W
Reserved	140	17F	64W
Ethernet MAL	180	1FF	128W
PLB4-to-OPB1 Bridge *(PPC440EPx only)*	200	20F	16W
Reserved	210	2FF	240W
DMA-to-PLB4 Controller	300	33F	64W
PLB4-to-OPB2 Bridge *(PPC440EPx only)*	340	34F	16W
OPB2-to-PLB4 Bridge	350	357	8W
Reserved	358	35F	8W
USB 2.0 Host OPB Master DCR *(PPC440EPx only)*	360	367	8W
Reserved	368	36F	8W
USB 2.0 Host OPB Slave DCR *(PPC440EPx only)*	370	373	4W
Reserved	374	37F	12W

Table 3-2. DCR Address Map (4KB of Device Configuration Registers)

Function	Start Address	End Address	Size
On Chip Memory (SRAM Controller)	380	38F	16W
Reserved	390	3FF	112W

Notes:

1. DCR addresses are 10 bits (1024 or 1K unique addresses). Each unique address represents a single 32-bit (word) register. One kiloword (1024W) equals 4KB (4096 B).

3.2.1 Device Control Registers

DCRs may be used to control various on-chip system functions, such as the operation of on-chip buses, peripherals, and certain processor core behaviors. The DCR access instructions are mtdcr (move to device control register) and mfdcr (move from device control register), which move data between GPRs and the DCRs. Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR.

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order)

Register	DCR Address or Offset (See Note)	Access	Description
Clocking and Power-on Reset (See Note)			
CPR0_CFGADDR	0x00C	R/W	Clock, Power-On, Reset Configuration Address Register
CPR0_CFGDATA	0x00D	R/W	Clock, Power-On, Reset Configuration Data Register
CPR0_CLKUPD	offset 0x0020	R/W	Clocking Update Register
CPR0_PLLC0	offset 0x0040	R/W	PLL Control Register
CPR0_PLLD0	offset 0x0060	R/W	PLL Divisor Register
CPR0_PRIMAD0	offset 0x0080	R/W	Primary Divisor Register A
CPR0_PRIMBD0	offset 0x00A0	R/W	Primary Divisor Register B
CPR0_OPBD0	offset 0x00C0	R/W	OPB Clock Divisor Register
CPR0_PERD0	offset 0x00E0	R/W	Peripheral Clock Divisor Register
CPR0_MALD	offset 0x0100	R/W	MAL Clock Divisor Register
CPR0_SPCID	offset 0x0120	R/W	Synchronous PCI Clock Divisor Register
CPR0_ICFG	offset 0x0140	R/W	Initial Configuration Register
System Device Control (See Note)			
SDR0_CFGADDR	0x00E	R/W	System DCR Configuration Address Register
SDR0_CFGDATA	0x00F	R/W	System DCR Configuration Data Register
SDR0_SDSTP0	offset 0x0020	R	Serial Device Strap Register 0
SDR0_SDSTP1	offset 0x0021	R	Serial Device Strap Register 1
SDR0_PINSTP	offset 0x0040	R	Pin Strapping Register
SDR0_SDCS0	offset 0x0060	R/W	Serial Device Controller Settings Register
SDR0_ECID0	offset 0x0080	R/W	Electronic Chip ID Register 0
SDR0_ECID1	offset 0x0081	R/W	Electronic Chip ID Register 1
SDR0_ECID2	offset 0x0082	R/W	Electronic Chip ID Register 2

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
SDR0_ECID3	offset 0x0083	R/W	Electronic Chip ID Register 3
SDR0_JTAG	offset 0x00C0	R/W	JTAG ID Register
SDR0_DDRCFG	offset 0x00E0	R/W	DDR Configuration Register
SDR0_EBC0	offset 0x0100	R/W	EBC Configuration Register
SDR0_UART0	offset 0x0120	R/W	UART Configuration Register 0
SDR0_UART1	offset 0x0121	R/W	UART Configuration Register 1
SDR0_UART2	offset 0x0122	R/W	UART Configuration Register 2
SDR0_UART3	offset 0x0123	R/W	UART Configuration Register 3
SDR0_CP440	offset 0x0180	R/W	440CPU Control Register
SDR0_SRST0	offset 0x0200	R/W	Individual Core Reset Control Register 0
SDR0_SRST1	offset 0x0201	R/W	Individual Core Reset Control Register 1
SDR0_SLPIPE0	offset 0x0220	R/W	PLB Slave Address Pipeline Disabling Register
SDR0_AMP0	offset 0x0240	R/W	Alternate PLB4 Master Priority Register
SDR0_AMP1	offset 0x0241	R/W	Alternate PLB3 Master Priority Register
SDR0_MIRQ0	offset 0x0260	R/W	Master Interrupt Request Register 0
SDR0_MIRQ1	offset 0x0261	R/W	Master Interrupt Request Register 1
SDR0_MALTBL	offset 0x0280	R/W	MAL Transmit Burst Length Register
SDR0_MALRBL	offset 0x02A0	R/W	MAL Receive Burst Length Register
SDR0_MALTBS	offset 0x02C0	R/W	MAL Transmit Bus Size Register
SDR0_MALRBS	offset 0x02E0	R/W	MAL Receive Bus Size Register
SDR0_PCI0	offset 0x0300	R/W	PCI Control Register
SDR0_USB2D0CR	offset 0x0320	R/W	USB2 Device Control Register
SDR0_USB2H0CR	offset 0x0340	R/W	USB2 Host Control Register
SDR0_CUST0	offset 0x4000	R/W	Register0 Reserved for Customer Use
SDR0_SDSTP2	offset 0x4001	R	Read Only Version of SDR0_CUST0
SDR0_CUST1	offset 0x4002	R/W	Register1 Reserved for Customer Use
SDR0_SDSTP3	offset 0x4003	R	Read Only Version of SDR0_CUST1
SDR0_PFC0	offset 0x4100	R/W	Pin Function Control Register 0
SDR0_PFC1	offset 0x4101	R/W	Pin Function Control Register 1
SDR0_PFC2	offset 0x4102	R/W	Ethernet PLL Configuration Register
SDR0_USB2PHY0CR	offset 0x4103	R/W	USB2PHY0 Control Register
SDR0_PFC4	offset 0x4104	R/W	GPIO Multiplexing Register
SDR0_MFR	offset 0x4300	R/W	Miscellaneous Function Register
SDR0_EMAC0RXST	offset 0x4301	R/W	EMAC0 RX Status Register
SDR0_EMAC0TXST	offset 0x4302	R/W	EMAC0 TX Status Register
SDR0_EMAC1RXST	offset 0x4303	R/W	EMAC1 RX Status Register
SDR0_EMAC1TXST	offset 0x4304	R/W	EMAC1 TX Status Register
SDR0_EMAC0REJCNT	offset 0x4305	R	EMAC0 RX Packet Reject Counter

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
SDR0_EMAC1REJCNT	offset 0x4306	R	EMAC1 RX Packet Reject Counter
SDR0_ICSRTX0	offset 0x4307	R/Clear	Interrupt Coalescing Status Register Transmit Channel 0
SDR0_ICSRTX1	offset 0x4308	R/Clear	Interrupt Coalescing Status Register Transmit Channel 1
SDR0_ICSRRX0	offset 0x4309	R/Clear	Interrupt Coalescing Status Register Receive Channel 0
SDR0_ICSRRX1	offset 0x430A	R/Clear	Interrupt Coalescing Status Register Receive Channel 1
SDR0_ICCRTX	offset 0x430B	R/W	Interrupt Coalescing Control Transmit Register
SDR0_ICCRRX	offset 0x430C	R/W	Interrupt Coalescing Control Receive Register
SDR0_ICTRTX0	offset 0x430D	R/W	Interrupt Coalescing Timer Register Transmit Channel 0
SDR0_ICTRTX1	offset 0x430E	R/W	Interrupt Coalescing Timer Register Transmit Channel 1
SDR0_ICTRRX0	offset 0x430F	R/W	Interrupt Coalescing Timer Register Receive Channel 0
SDR0_ICTRRX1	offset 0x4310	R/W	Interrupt Coalescing Timer Register Receive Channel 1
SDR0_CRYP0	offset 0x4500	R/W	Security Function Configuration Register
SDR0_USB2H0ST	offset 0x4600	Read only	USB2 Host Status Register
DDR SDRAM Controller (See Note)			
DDR0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
DDR0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register
DDR0_00	offset 0x00	See register	DDR-SDRAM Register
DDR0_01	offset 0x01	See register	DDR-SDRAM Register
DDR0_02	offset 0x02	See register	DDR-SDRAM Register
DDR0_03	offset 0x03	R/W	DDR-SDRAM Register
DDR0_04	offset 0x04	R/W	DDR-SDRAM Register
DDR0_05	offset 0x05	R/W	DDR-SDRAM Register
DDR0_06	offset 0x06	R/W	DDR-SDRAM Register
DDR0_07	offset 0x07	See register	DDR-SDRAM Register
DDR0_08	offset 0x08	R/W	DDR-SDRAM Register
DDR0_09	offset 0x09	R/W	DDR-SDRAM Register
DDR0_10	offset 0x0A	See register	DDR-SDRAM Register
DDR0_11	offset 0x0B	R/W	DDR-SDRAM Register
DDR0_12	offset 0x0C	R/W	DDR-SDRAM Register
DDR0_14	offset 0x0E	R/W	DDR-SDRAM Register
DDR0_17	offset 0x11	R/W	DDR-SDRAM Register
DDR0_18	offset 0x12	R/W	DDR-SDRAM Register
DDR0_19	offset 0x13	R/W	DDR-SDRAM Register
DDR0_20	offset 0x14	R/W	DDR-SDRAM Register
DDR0_21	offset 0x15	R/W	DDR-SDRAM Register
DDR0_22	offset 0x16	R/W	DDR-SDRAM Register
DDR0_23	offset 0x17	See register	DDR-SDRAM Register
DDR0_24	offset 0x18	R/W	DDR-SDRAM Register

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
DDR0_25	offset 0x19	R/W	DDR-SDRAM Register
DDR0_26	offset 0x1A	R/W	DDR-SDRAM Register
DDR0_27	offset 0x1B	R/W	DDR-SDRAM Register
DDR0_28	offset 0x1C	R/W	DDR-SDRAM Register
DDR0_31	offset 0x1F	R/W	DDR-SDRAM Register
DDR0_32	offset 0x20	Read only	DDR-SDRAM Register
DDR0_33	offset 0x21	Read only	DDR-SDRAM Register
DDR0_34	offset 0x22	Read only	DDR-SDRAM Register
DDR0_35	offset 0x23	Read only	DDR-SDRAM Register
DDR0_36	offset 0x24	Read only	DDR-SDRAM Register
DDR0_37	offset 0x25	Read only	DDR-SDRAM Register
DDR0_38	offset 0x26	Read only	DDR-SDRAM Register
DDR0_39	offset 0x27	Read only	DDR-SDRAM Register
DDR0_40	offset 0x28	Read only	DDR-SDRAM Register
DDR0_41	offset 0x29	Read only	DDR-SDRAM Register
DDR0_42	offset 0x2A	R/W	DDR-SDRAM Register
DDR0_43	offset 0x2B	R/W	DDR-SDRAM Register
DDR0_44	offset 0x2C	R/W	DDR-SDRAM Register
External Bus Controller (See Note)			
EBC0_CFGADDR	0x012	R/W	Peripheral Bank Address Register
EBC0_CFGDATA	0x013	R/W	Peripheral Bank Data Register
EBC0_B0CR	offset 0x0000	R/W	Peripheral Bank 0 Configuration Register
EBC0_B1CR	offset 0x0001	R/W	Peripheral Bank 1 Configuration Register
EBC0_B2CR	offset 0x0002	R/W	Peripheral Bank 2 Configuration Register
EBC0_B3CR	offset 0x0003	R/W	Peripheral Bank 3 Configuration Register
EBC0_B4CR	offset 0x0004	R/W	Peripheral Bank 4 Configuration Register
EBC0_B5CR	offset 0x0005	R/W	Peripheral Bank 5 Configuration Register
EBC0_B0AP	offset 0x0010	R/W	Peripheral Bank 0 Access Parameters
EBC0_B1AP	offset 0x0011	R/W	Peripheral Bank 1 Access Parameters
EBC0_B2AP	offset 0x0012	R/W	Peripheral Bank 2 Access Parameters
EBC0_B3AP	offset 0x0013	R/W	Peripheral Bank 3 Access Parameters
EBC0_B4AP	offset 0x0014	R/W	Peripheral Bank 4 Access Parameters
EBC0_B5AP	offset 0x0015	R/W	Peripheral Bank 5 Access Parameters
EBC0_BEAR	offset 0x0020	R/W	Peripheral Bus Error Address Register
EBC0_BESR0	offset 0x0021	R/W	Peripheral Bus Error Status Register 0
EBC0_CFG	offset 0x0023	R/W	External Peripheral Control Register
PLB4 to PLB3 Bridge			
P4P3BO0_BESR0	0x020	R/Clear	PLB4 to PLB3 Bridge Error Status 0

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
P4P3BO0_BEARL	0x022	R/Clear	PLB4 to PLB3 Bridge Error Address Low
P4P3BO0_BEARH	0x023	R/Clear	PLB4 to PLB3 Bridge Error Address High
P4P3BO0_BESR1	0x024	R/Clear	PLB4 to PLB3 Bridge Error Status 1
P4P3BO0_CFG	0x026	R/Clear/Set	PLB4 to PLB3 Bridge Configuration
P4P3BO0_PICR	0x027	R/Clear/Set	PLB4 to PLB3 Bridge Priority
P4P3BO0_PEIR	0x028	R/Clear/Set	PLB4 to PLB3 Bridge Parity Error Interrupt
P4P3BO0_REVID	0x02A	R/Clear	PLB4 to PLB3 Bridge Revision ID
PLB3 to PLB4 Bridge			
P3P4BI0_BESR0	0x030	R/Clear	PLB3 to PLB4 Bridge Error Status 0
P3P4BI0_BEARL	0x032	R/Clear	PLB3 to PLB4 Bridge Error Address Low
P3P4BI0_BEARH	0x033	R/Clear	PLB3 to PLB4 Bridge Error Address High
P3P4BI0_BESR1	0x034	R/Clear	PLB3 to PLB4 Bridge Error Status 1
P3P4BI0_CFG	0x036	R/Clear/Set	PLB3 to PLB4 Bridge Configuration
P3P4BI0_PICR	0x037	R/Clear/Set	PLB3 to PLB4 Bridge Priority
P3P4BI0_PEIR	0x038	R/Clear/Set	PLB3 to PLB4 Bridge Parity Error Interrupt
P3P4BI0_REVID	0x03A	R/Clear	PLB3 to PLB4 Bridge Revision ID
PLB3 Arbiter Registers			
PLB3A0_REVID	0x072	R	PLB3 Arbiter Revision ID Register
PLB3A0_BESR	0x074	R/Clear	PLB3 Bus Error Status Register
PLB3A0_BEAR	0x076	R/W	PLB3 Bus Error Address Register
PLB3A0_ACR	0x077	R/W	PLB3 Arbiter Control Register
PLB4 Arbiter Registers			
PLB4A0_REVID	0x080	R	PLB4 Crossbar ID/Revision Register
PLB4A0_ACR	0x081	R/W	PLB4A0 Arbiter Control Register
PLB4A0_ESRL	0x082	R/Clear	PLB4A0 Error Status Register Low
	0x083		Reserved
PLB4A0_EARL	0x084	R	PLB4A0 Error Address Register Low
PLB4A0_EARH	0x085	R	PLB4A0 Error Address Register High
PLB4A0_ESRL*	0x086	Set(W)	PLB4A0 Error Status Register Low (*reserved for diagnostic use only)
PLB4A0_ESRH*	0x087	Set(W)	PLB4A0 Error Status Register High (*reserved for diagnostic use only)
PLB4A0_CCR	0x088	R/W	PLB4 Crossbar Control Register
PLB4A1_ACR	0x089	R/W	PLB4A1 Arbiter Control Register
PLB4A1_ESRL	0x08A	R/Clear	PLB4A1 Error Status Register Low
	0x08B		Reserved
PLB4A1_EARL	0x08C	R	PLB4A1 Error Address Register Low
PLB4A1_EARH	0x08D	R	PLB4A1 Error Address Register High

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
PLB4A1_ESRL*	0x08E	Set(W)	PLB4A1 Error Status Register Low (*reserved for diagnostic use only)
PLB4A1_ESRH*	0x08F	Set(W)	PLB4A1 Error Status Register High (*reserved for diagnostic use only)
PLB3 to OPB Bridge			
PLB32OPB0_BESR0	0x090	R/Clear	PLB 3 to OPB Bridge Error Status Register 0
PLB32OPB0_BEAR	0x092	R	PLB 3 to OPB Bridge Error Address Register
PLB32OPB0_REVID	0x093	R	PLB 3 to OPB Bridge Revision ID Register
PLB32OPB0_BESR1	0x094	R/Clear	PLB 3 to OPB Bridge Error Status Register 1
Clock and Power Management 0			
CPM0_ER	0x0B0	R/W	CPM Enable Register
CPM0_FR	0x0B1	R/W	CPM Force Register
CPM0_SR	0x0B2	R/W	CPM Status Register
Universal Interrupt Controller 0			
UIC0_SR	0x0C0	R/Clear	UIC 0 Status Register
UIC0_SRS	0x0C1	W/Set	UIC 0 Status Register Set (reserved for debug only)
UIC0_ER	0x0C2	R/W	UIC 0 Enable Register
UIC0_CR	0x0C3	R/W	UIC 0 Critical Register
UIC0_PR	0x0C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x0C5	R/W	UIC 0 Triggering Register
UIC0_MSR	0x0C6	R	UIC 0 Masked Status Register
UIC0_VR	0x0C7	R	UIC 0 Vector Register
UIC0_VCR	0x0C8	W	UIC 0 Vector Configuration Register
Universal Interrupt Controller 1 *(PPC440EPx only)*			
UIC1_SR	0x0D0	R/Clear	UIC 1 Status Register
UIC1_SRS	0x0D1	W/Set	UIC 1 Status Register Set (reserved for debug only)
UIC1_ER	0x0D2	R/W	UIC 1 Enable Register
UIC1_CR	0x0D3	R/W	UIC 1 Critical Register
UIC1_PR	0x0D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x0D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x0D6	R	UIC 1 Masked Status Register
UIC1_VR	0x0D7	R	UIC 1 Vector Register
UIC1_VCR	0x0D8	W	UIC 1 Vector Configuration Register
Universal Interrupt Controller 2 *(PPC440EPx only)*			
UIC2_SR	0x0E0	R/Clear	UIC 2 Status Register
UIC2_SRS	0x0E1	W/Set	UIC 2 Status Register Set (reserved for debug only)
UIC2_ER	0x0E2	R/W	UIC 2 Enable Register

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
UIC2_CR	0x0E3	R/W	UIC 2 Critical Register
UIC2_PR	0x0E4	R/W	UIC 2 Polarity Register
UIC2_TR	0x0E5	R/W	UIC 2 Triggering Register
UIC2_MSR	0x0E6	R	UIC 2 Masked Status Register
UIC2_VR	0x0E7	R	UIC 2 Vector Register
UIC2_VCR	0x0E8	W	UIC 2 Vector Configuration Register
Clock and Power Management 1			
CPM1_ER	0x0F0	R/W	CPM 1 Enable Register
CPM1_FR	0x0F1	R/W	CPM 1 Force Register
CPM1_SR	0x0F2	R/W	CPM 1 Status Register
DMA to PLB3 Controller			
DMA2P30_CR0	0x100	R/W	DMA to PLB 3 Channel Control Register 0
DMA2P30_CT0	0x101	R/W	DMA to PLB 3 Count Register 0
DMA2P30_DA0	0x102	R/W	DMA to PLB 3 Destination Address Register 0
DMA2P30_SA0	0x103	R/W	DMA to PLB 3 Source Address Register 0
DMA2P30_SG0	0x104	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 0
DMA2P30_SC0	0x107	R/W	DMA to PLB 3 Subchannel ID Register 0
DMA2P30_CR1	0x108	R/W	DMA to PLB 3 Channel Control Register 1
DMA2P30_CT1	0x109	R/W	DMA to PLB 3 Count Register 1
DMA2P30_DA1	0x10A	R/W	DMA to PLB 3 Destination Address Register 1
DMA2P30_SA1	0x10B	R/W	DMA to PLB 3 Source Address Register 1
DMA2P30_SG1	0x10C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 1
DMA2P30_SC1	0x10F	R/W	DMA to PLB 3 Subchannel ID Register 1
DMA2P30_CR2	0x110	R/W	DMA to PLB 3 Channel Control Register 2
DMA2P30_CT2	0x111	R/W	DMA to PLB 3 Count Register 2
DMA2P30_DA2	0x112	R/W	DMA to PLB 3 Destination Address Register 2
DMA2P30_SA2	0x113	R/W	DMA to PLB 3 Source Address Register 2
DMA2P30_SG2	0x114	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 2
DMA2P30_SC2	0x117	R/W	DMA to PLB 3 Subchannel ID Register 2
DMA2P30_CR3	0x118	R/W	DMA to PLB 3 Channel Control Register 3
DMA2P30_CT3	0x119	R/W	DMA to PLB 3 Count Register 3
DMA2P30_DA3	0x11A	R/W	DMA to PLB 3 Destination Address Register 3
DMA2P30_SA3	0x11B	R/W	DMA to PLB 3 Source Address Register 3
DMA2P30_SG3	0x11C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 3
DMA2P30_SC3	0x11F	R/W	DMA to PLB 3 Subchannel ID Register 3
DMA2P30_SR	0x120	R/W	DMA to PLB 3 Status Register
DMA2P30_SGC	0x123	R/W	DMA to PLB 3 Scatter/Gather Command Register
DMA2P30_ADR	0x124	R/W	DMA to PLB 3 Address Decode Register

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
DMA2P30_SLP	0x125	R/W	DMA to PLB 3 Sleep Mode Register
DMA2P30_POL	0x126	R/W	DMA to PLB 3 Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	MAL Error Status Register
MAL0_IER	0x182	R/W	MAL Interrupt Enable Register
MAL0_TXCASR	0x184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_TXTATTRR	0x188	R/W	MAL Tx PLB Attribute Register
MAL0_TXBADDR	0x189	R/W	MAL Tx Descriptor Base Address Register
MAL0_RXCASR	0x190	R/W	MAL Rx Channel Active Register (Set)
MAL0_RXCARR	0x191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_RXTTATTR	0x194	R/W	MAL Rx PLB Attribute Register
MAL0_RXBADDR	0x195	R/W	MAL Rx Descriptor Base Address Register
MAL0_TXCTP0R	0x1A0	R/W	MAL Tx Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	MAL Tx Channel 1 Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	MAL Rx Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x1C1	R/W	MAL Rx Channel 1 Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	MAL Rx Channel 0 Buffer Size Register
MAL0_RCBS1	0x1E1	R/W	MAL Rx Channel 1 Buffer Size Register
PLB4 to OPB Bridge 0 (for USB 2.0 Device), *(PPC440EPx only)*			
PLB42OPB0_BESR0	0x200	R/Clear	PLB4 to OPB Bridge Error Status Register 0
PLB42OPB0_BEARL	0x202	R	PLB4 to OPB Bridge Error Address Register
PLB42OPB0_BEARH	0x203	R	PLB4 to OPB Bridge Error Address Register
PLB42OPB0_BESR1	0x204	R/Clear	PLB4 to OPB Bridge Error Status Register 1
PLB42OPB0_CFG	0x206	R/Clear	PLB4 to OPB Bridge Configuration Register
PLB42OPB0_LATENCY	0x208	R/Clear	PLB4 to OPB Bridge Burst Latency Timer
PLB42OPB0_REVID	0x20A	R	PLB4 to OPB Bridge Revision ID Register
DMA to PLB4 Controller			
DMA2P40_CR0	0x300	R/W	DMA to PLB 4 Channel Control Register 0
DMA2P40_CTC0	0x301	R/W	DMA to PLB 4 Count and Control Register 0
DMA2P40_SAH0	0x302	R/W	DMA to PLB 4 Source Address High Register 0
DMA2P40_SAL0	0x303	R/W	DMA to PLB 4 Source Address Low Register 0
DMA2P40_DAH0	0x304	R/W	DMA to PLB 4 Destination Address High Register 0

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
DMA2P40_DAL0	0x305	R/W	DMA to PLB 4 Destination Address Low Register 0
DMA2P40_SGH0	0x306	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 0
DMA2P40_SGL0	0x307	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 0
DMA2P40_CR1	0x308	R/W	DMA to PLB 4 Channel Control Register 1
DMA2P40_CTC1	0x309	R/W	DMA to PLB 4 Count and Control Register 1
DMA2P40_SAH1	0x30A	R/W	DMA to PLB 4 Source Address High Register 1
DMA2P40_SAL1	0x30B	R/W	DMA to PLB 4 Source Address Low Register 1
DMA2P40_DAH1	0x30C	R/W	DMA to PLB 4 Destination Address High Register 1
DMA2P40_DAL1	0x30D	R/W	DMA to PLB 4 Destination Address Low Register 1
DMA2P40_SGH1	0x30E	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 1
DMA2P40_SGL1	0x30F	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 1
DMA2P40_CR2	0x310	R/W	DMA to PLB 4 Channel Control Register 2
DMA2P40_CTC2	0x311	R/W	DMA to PLB 4 Count and Control Register 2
DMA2P40_SAH2	0x312	R/W	DMA to PLB 4 Source Address High Register 2
DMA2P40_SAL2	0x313	R/W	DMA to PLB 4 Source Address Low Register 2
DMA2P40_DAH2	0x314	R/W	DMA to PLB 4 Destination Address High Register 2
DMA2P40_DAL2	0x315	R/W	DMA to PLB 4 Destination Address Low Register 2
DMA2P40_SGH2	0x316	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 2
DMA2P40_SGL2	0x317	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 2
DMA2P40_CR3	0x318	R/W	DMA to PLB 4 Channel Control Register 3
DMA2P40_CTC3	0x319	R/W	DMA to PLB 4 Count and Control Register 3
DMA2P40_SAH3	0x31A	R/W	DMA to PLB 4 Source Address High Register 3
DMA2P40_SAL3	0x31B	R/W	DMA to PLB 4 Source Address Low Register 3
DMA2P40_DAH3	0x31C	R/W	DMA to PLB 4 Destination Address High Register 3
DMA2P40_DAL3	0x31D	R/W	DMA to PLB 4 Destination Address Low Register 3
DMA2P40_SGH3	0x31E	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 3
DMA2P40_SGL3	0x31F	R/W	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 3
DMA2P40_SR	0x320	R/W	DMA to PLB 4 Status Register
DMA2P40_SGC	0x323	R/W	DMA to PLB 4 Scatter/Gather Command Register
DMA2P40_SLP	0x325	R/W	DMA to PLB 4 Sleep Mode Register
DMA2P40_POL	0x326	R/W	DMA to PLB 4 Polarity Configuration Register
PLB4 to OPB Bridge 1 (for USB 2.0 Host) *(PPC440EPx only)*			
PLB42OPB1_BESR0	0x340	R/Clear	PLB4 to OPB Bridge Error Status Register 0
PLB42OPB1_BEARL	0x342	R	PLB4 to OPB Bridge Error Address Register
PLB42OPB1_BEARH	0x343	R	PLB4 to OPB Bridge Error Address Register
PLB42OPB1_BESR1	0x344	R/Clear	PLB4 to OPB Bridge Error Status Register 1
PLB42OPB1_CFG	0x346	R/Clear	PLB4 to OPB Bridge Error Configuration Register
PLB42OPB1_LATENCY	0x348	R/Clear	PLB4 to OPB Bridge Burst Latency Timer

Table 3-3. PPC440EPx/GRx Device Control Registers (in Number or Offset order) (continued)

Register	DCR Address or Offset (See Note)	Access	Description
PLB42OPB1_REVID	0x34A	R	PLB4 to OPB Bridge Revision ID Register
OPB2 to PLB4 Bridge *(PPC440EPx only)*			
OPB2PLB40_BCTRL	0x350	R/W	OPB to PLB4 Bridge Control Register
OPB2PLB40_BSTAT	0x351	R	OPB to PLB4 Bridge Status Register
OPB2PLB40_BEARL	0x352	R	OPB to PLB4 Error Address Register Low
OPB2PLB40_BEARH	0x353	R	OPB to PLB4 Error Address Register High
OPB2PLB40_REVID	0x354	R	OPB to PLB4 Bridge Revision ID Register
USB 2.0 Host *(PPC440EPx only)*			
USB2HMDCR0_BCNTL	0x360	R/W	OPB Master Interface Bridge Control
USB2HMDCR0_BSTAT	0x361	R/W	OPB Master Interface Bridge Status
USB2HMDCR0_LAT	0x366	R/W	OPB Master Interface Bridge Latency
USB2HMDCR0_REVID	0x367	R	OPB Master Interface Bridge Revision ID
USB2HSDCR0_LAT	0x370	R/W	OPB Slave Interface Bridge Control
USB2HSDCR0_REVID	0x373	R	OPB Slave Interface Bridge Revision ID
SRAM Controller			
SRAM0_SB0CR	0x380	R/W	SRAM Bank 0 Configuration Register
SRAM0_BEAR	0x384	R/W	SRAM Bus Error Address Register
SRAM0_BESR0	0x385	R/W	Bus Error Status Register 0
SRAM0_BESR1	0x386	R/W	Bus Error Status Register 1
SRAM0_PMEG	0x387	R/W	Power Management Register
SRAM0_CID	0x388	R	Core ID Register
SRAM0_REVID	0x389	R	Revision ID Register
SRAM0_DPC	0x38A	R/W	Data Parity Check Register

Note: The CPR0_, SDR0_, DDR0_, and EBC0_ register groups include both registers that are accessed through their unique DCR numbers and registers that are accessed through their offset values. Each of these groups has its uniquely-numbered xxxx_CFGADDR and xxxx_CFGDATA registers that are used to indirectly address the other registers in the group through their offset values. *Section 22.6 EBC Registers* on page 574 contains a detailed example of indirect addressing.

3.2.2 Memory Mapped Registers

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses.

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order)

Register	Address	Access	Description
OPB Arbiter 1 (attached to 128-bit PLB) *(PPC440EPx Only)*			
OPBA1_PR	0x0 E000 0000	R/W	OPB Arbiter Priority Register
OPBA1_CR	0x0 E000 0001	R/W	OPB Arbiter Control Register
USB 2.0 Device *(PPC440EPx Only)*			
USB2D0_INTRIN	0x0 E000 0100	R	Interrupt register for Endpoint 0 plus IN Endpoints 1 to 3
USB2D0_POWER	0x0 E000 0102	R/W	Power management register
USB2D0_FADDR	0x0 E000 0103	R/W	Function address register
USB2D0_INTRINE	0x0 E000 0104	R/W	Interrupt enable register for USB2D0_INTRIN
USB2D0_INTROUT	0x0 E000 0106	R	Interrupt register for OUT Endpoints 1 to 3
USB2D0_INTRUSBE	0x0 E000 0108	R/W	Interrupt enable register for USB2D0_INTRUSB
USB2D0_INTRUSB	0x0 E000 0109	R	Interrupt register for common USB interrupts
USB2D0_INTROUTE	0x0 E000 010A	R/W	Interrupt enable register for IntrOut
USB2D0_TSTMODE	0x0 E000 010C	R/W	Enables the USB 2.0 test modes
USB2D0_INDEX	0x0 E000 010D	R/W	Index register for selecting the Endpoint status/control registers
USB2D0_FRAME	0x0 E000 010E	R	Frame number
USB2D0_INCSR0	0x0 E000 0110	R/W	Control Status register for Endpoint 0. (Index register set to select Endpoint 0)
USB2D0_INCSR	0x0 E000 0110	R/W	Control Status register for IN Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_INMAXP	0x0 E000 0112	R/W	Maximum packet size for IN Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_OUTCSR	0x0 E000 0114	R/W	Control Status register for OUT Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_OUTMAXP	0x0 E000 0116	R/W	Maximum packet size for OUT Endpoint. (Index register set to select Endpoints 1–3)
USB2D0_OUTCOUNT0	0x0 5000 011A	R	Number of received bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)
USB2D0_OUTCOUNT	0x0 E000 011A	R	Number of bytes in OUT Endpoint FIFO. (Index register set to select Endpoints 1–3)
OPB Arbiter 2 (attached to 128-bit PLB) *(PPC440EPx Only)*			
OPBA2_PR	0x0 E000 0200	R/W	OPB Arbiter Priority Register
OPBA2_CR	0x0 E000 0201	R/W	OPB Arbiter Control Register
USB 2.0 Host *(PPC440EPx Only)*			
EHCI0_HCCAPBASE	0xE000 0300	R	Version/Capability Length
EHCI0_HCSPARAMS	0xE000 0304	R	Structural Parameters
EHCI0_HCCPARAMS	0xE000 0308	R	Capability Parameters

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
EHCI0_USBCMD	0x0 E000 0310	R/W	USB Command
EHCI0_USBSTS	0x0 E000 0314	R/W	USB Status
EHCI0_USBINTR	0x0 E000 0318	R/W	USB Interrupt Enable
EHCI0_FRINDEX	0x0 E000 031C	R/W	EHCI Frame Index
EHCI0_CRTLSEG	0x0 E000 0320	R/W	Control Data Structure Segment
EHCI0_PRDLISTB	0x0 E000 0324	R/W	Periodic Frame List Base Address
EHCI0_ASYNCLSTA	0x0 E000 0328	R/W	Current Asynchronous List Address
EHCI0_CFGFLAG	0x0 E000 0350	R/W	Configure Flag
EHCI0_PORTSC	0x0 E000 0354	R/W	Port Status and Control
EHCI0_INSNREG0	0x0 E000 0390	R/W	Programmable Microframe Base Value
EHCI0_INSNREG1	0x0 E000 0394	R/W	Programmable Packet Buffer Threshold
EHCI0_INSNREG2	0x0 E000 0398	R/W	Programmable Packet Buffer Depth
EHCI0_INSNREG3	0x0 E000 039C	R/W	Break Memory Transfer
EHCI0_INSNREG4	0x0 E000 03A0	R/W	Debug Only
OHCI0_HCREV	0x0 E000 0400	R	Revision
OHCI0_HCCTRL	0x0 E000 0404	R/W	Control
OHCI0_HCCMDSTS	0x0 E000 0408	R/W	Command Status
OHCI0_HCINTSTS	0x0 E000 040C	R/W	Interrupt Status
OHCI0_HCINTE	0x0 E000 0410	R/W	Interrupt Enable
OHCI0_HCINTDIS	0x0 E000 0414	R/W	Interrupt Disable
OHCI0_HCHCCA	0x0 E000 0418	R/W	Host Controller Communications Area
OHCI0_HCPCED	0x0 E000 041C	R/W	Periodic Current Endpoint Descriptor
OHCI0_HCCHED	0x0 E000 0420	R/W	Control Head Endpoint Descriptor
OHCI0_HCCTRLCED	0x0 E000 0424	R/W	Control Current Endpoint Descriptor
OHCI0_HCBULKHED	0x0 E000 0428	R/W	Bulk Head Endpoint Descriptor
OHCI0_HCBULKCED	0x0 E000 042C	R/W	Bulk Current Endpoint Descriptor
OHCI0_HCDHEAD	0x0 E000 0430	R	Done Head Transfer Descriptor
OHCI0_HCFMINT	0x0 E000 0434	R/W	Frame Interval
OHCI0_HCFMREM	0x0 E000 0438	R	Frame Remaining
OHCI0_HCFMNUM	0x0 E000 043C	R	Frame Number
OHCI0_HCPRDSTRT	0x0 E000 0440	R/W	Periodic Start
OHCI0_HCLSTHRES	0x0 E000 0444	R/W	Low Speed Threahold
OHCI0_HCRHDESCA	0x0 E000 0448	R/W	Root Hub Descriptor A
OHCI0_HCRHDESCB	0x0 E000 044C	R/W	Root Hub Descriptor B
OHCI0_HCRHSTS	0x0 E000 0450	R/W	Root Hub Status
OHCI0_HCPRSTS	0x0 E000 0454	R/W	Root Hub Port Status
Security Function			
CRYP0_PE_CTLST	0xE010 0000	R/W	PE Control/Status

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
CRYP0_PE_SOURCE	0xE010 0004	R/W	PE Source Address
CRYP0_PE_DEST	0xE010 0008	R/W	PE Destination Address
CRYP0_PE_SA	0xE010 000C	R/W	PE SA Address
CRYP0_PE_LENGTH	0xE010 0010	R/W	PE Length
CRYP0_SA_CMD0	0xE010 0014	R	See SA Command 0 Word
CRYP0_SA_CMD1	0xE010 0018	R	See SA Command 1 Word
CRYP0_PE_DMA_CF	0xE010 0040	See register	PE DMA Configuration
CRYP0_PE_DMA_ST	0xE010 0044	R	PE DMA Status
CRYP0_PE_PDR_BA	0xE010 0048	R/W	PE Packet Descriptor Ring Base Address
CRYP0_PE_RDR_BA	0xE010 004C	R/W	PE Result Descriptor Ring Base Address
CRYP0_PE_RING_S	0xE010 0050	R/W	PE Ring Size and Offset
CRYP0_PE_RING_P	0xE010 0054	R/W	PE Ring Poll
CRYP0_PE_I_RING	0xE010 0058	R	PE Internal Ring Status
CRYP0_PE_E_RING	0xE010 005C	R	PE External Ring Status
CRYP0_PE_IO_THR	0xE010 0060	R/W	PE I/O Threshold
CRYP0_PE_GATH	0xE010 0064	R/W	PE Gather Particle Ring Base Address
CRYP0_PE_SCAT	0xE010 0068	R/W	PE Scatter Particle Ring Base Address
CRYP0_PE_PT_S	0xE010 006C	R/W	PE Particle Ring Size
CRYP0_PE_PT_CFG	0xE010 0070	R/W	PE Particle Ring Configuration
CRYP0_PE_PR_SCA	0xE010 0500	R/W	PE Particle Descriptor Source Address
CRYP0_PE_PR_SCC	0xE010 0504	R/W	PE Particle Descriptor Source Control
CRYP0_PE_PR_DTA	0xE010 0580	R/W	PE Particle Descriptor Destination Address
CRYP0_PE_PR_DTC	0xE010 0584	R/W	PE Particle Descriptor Destination Control
CRYP0_SA_CMD_0	0xE011 0600	W	SA Command 0 Word
CRYP0_SA_CMD_1	0xE011 0604	W	SA Command 1 Word
CRYP0_SA_KEY1_L	0xE011 0610	W	SA Key 1 Low
CRYP0_SA_KEY1_H	0xE011 0614	W	SA Key 1 High
CRYP0_SA_KEY2_L	0xE011 0618	W	SA Key 2 Low
CRYP0_SA_KEY2_H	0xE011 061C	W	SA Key 2 High
CRYP0_SA_KEY3_L	0xE011 0620	W	SA Key 3 Low
CRYP0_SA_KEY3_H	0xE011 0624	W	SA Key 3 High
CRYP0_SA_KEY4_L	0xE011 0628	W	SA Key 4 Low
CRYP0_SA_KEY4_H	0xE011 062C	W	SA Key 4 High
CRYP0_SA_IH_D0	0xE011 0630	R/W	SA Inner Hash Digest 0
CRYP0_SA_IH_D1	0xE011 0634	R/W	SA Inner Hash Digest 1
CRYP0_SA_IH_D2	0xE011 0638	R/W	SA Inner Hash Digest 2
CRYP0_SA_IH_D3	0xE011 063C	R/W	SA Inner Hash Digest 3
CRYP0_SA_IH_D4	0xE011 0640	R/W	SA Inner Hash Digest 4

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
CRYP0_SA_OH_D0	0xE011 0644	W	SA Outer Hash Digest 0
CRYP0_SA_OH_D1	0xE011 0648	W	SA Outer Hash Digest 1
CRYP0_SA_OH_D2	0xE011 064C	W	SA Outer Hash Digest 2
CRYP0_SA_OH_D3	0xE011 0650	W	SA Outer Hash Digest 3
CRYP0_SA_OH_D4	0xE011 0654	W	SA Outer Hash Digest 4
CRYP0_SA_SPI	0xE011 0658	R/W	SA IPsec SPI
CRYP0_SA_SEQ	0xE011 065C	R/W	SA IPsec Sequence Number
CRYP0_SA_SEQMKL	0xE011 0660	R/W	SA IPsec Sequence Number Mask Low
CRYP0_SA_SEQMKH	0xE011 0664	R/W	SA IPsec Sequence Number Mask High
CRYP0_SA_NONCE	0xE011 0668	R/W	SA Nonce Value
CRYP0_SA_PNTR	0xE011 066C	R/W	SA Pointer
CRYP0_SA_ARC4IJ	0xE011 0670	R/W	SA ARC4 i and j Pointer
CRYP0_SA_ARC4SB	0xE011 0674	R/W	SA ARC4 State Address Pointer
CRYP0_SA_IV_0	0xE011 06C0	R/W	SA Initialization Vector 0
CRYP0_SA_IV_1	0xE011 06C4	R/W	SA Initialization Vector 1
CRYP0_SA_IV_2	0xE011 06C8	R/W	SA Initialization Vector 2
CRYP0_SA_IV_3	0xE011 06CC	R/W	SA Initialization Vector 3
CRYP0_SA_HASH_B	0xE011 06D0	R/W	SA Hash Byte Count
CRYP0_SA_IH_0	0xE011 06D4	R/W	SA Inner Hash 0 (mirror of CRYP0_SA_IH_D0)
CRYP0_SA_IH_1	0xE011 06D8	R/W	SA Inner Hash 1 (mirror of CRYP0_SA_IH_D1)
CRYP0_SA_IH_2	0xE011 06DC	R/W	SA Inner Hash 2 (mirror of CRYP0_SA_IH_D2)
CRYP0_SA_IH_3	0xE011 06E0	R/W	SA Inner Hash 3 (mirror of CRYP0_SA_IH_D3)
CRYP0_SA_IH_4	0xE011 06E4	R/W	SA Inner Hash 4 (mirror of CRYP0_SA_IH_D4)
CRYP0_SA_ICV_0	0xE011 06E8	R	SA ICV 0—HMAC result (outbound and inbound)
CRYP0_SA_ICV_1	0xE011 06EC	R	SA ICV 1—HMAC result (outbound and inbound)
CRYP0_SA_ICV_2	0xE011 06F0	R	SA ICV 2—HMAC result (outbound and inbound)
CRYP0_SA_ICV_3	0xE011 06F4	R	SA ICV 3—HMAC result (outbound and inbound)
CRYP0_SA_ICV_4	0xE011 06F8	R	SA ICV 4—HMAC result (outbound and inbound)
CRYP0_TRNG_DATA	0xE012 0100	R/W	TRNG Output
CRYP0_TRNG_STAT	0xE012 0104	R	TRNG Status
CRYP0_TRNG_CTRL	0xE012 0108	R/W	TRNG Test Control
CRYP0_TRNG_ENTA	0xE012 010C	R/W	TRNG Test Entropy A
CRYP0_TRNG_ENTB	0xE012 0110	R/W	TRNG Test Entropy B
CRYP0_TRNG_X0	0xE012 0114	R/W	TRNG Test Seed 0
CRYP0_TRNG_X1	0xE012 0118	R/W	TRNG Test Seed 1
CRYP0_TRNG_X2	0xE012 011C	R/W	TRNG Test Seed 2
CRYP0_TRNG_CNTR	0xE012 0120	R/W	TRNG Counter
CRYP0_TRNG_ALRM	0xE012 0124	R	TRNG Alarm Count

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
CRYP0_TRNG_CFG	0xE012 0128	R/W	TRNG Configuration
CRYP0_TRNG_LF0L	0xE012 012C	R	TRNG Test Read of LFSR 0 Low
CRYP0_TRNG_LF0H	0xE012 0130	R	TRNG Test Read of LFSR0 High
CRYP0_TRNG_LF1L	0xE012 0134	R	TRNG Test Read of LFSR1 Low
CRYP0_TRNG_LF1H	0xE012 0138	R	TRNG Test Read of LFSR1 High
CRYP0_TRNG_K0_L	0xE012 013C	W	TRNG Triple DES Key 0 Low
CRYP0_TRNG_K0_H	0xE012 0140	W	TRNG Triple DES Key 0 High
CRYP0_TRNG_K1_L	0xE012 0144	W	TRNG Triple DES Key 1 Low
CRYP0_TRNG_K1_H	0xE012 0148	W	TRNG Triple DES Key 1 High
CRYP0_TRNG_IV_L	0xE012 014C	W	TRNG Initialization Vector Low
CRYP0_TRNG_IV_H	0xE012 0150	W	TRNG Initialization Vector High
CRYP0_PKA_A_PTR	0xE013 0800	R/W	PKA A Vector Address
CRYP0_PKA_B_PTR	0xE013 0804	R/W	PKA B Vector Address
CRYP0_PKA_C_PTR	0xE013 0808	R/W	PKA C Vector Address
CRYP0_PKA_D_PTR	0xE013 080C	R/W	PKA D Vector Address
CRYP0_PKA_A_LEN	0xE013 0810	R/W	PKA A Vector Length
CRYP0_PKA_B_LEN	0xE013 0814	R/W	PKA B Vector Length
CRYP0_PKA_SHIFT	0xE013 0818	R/W	PKA Shift
CRYP0_PKA_FUNC	0xE013 081C	R/W	PKA Function Code
CRYP0_PKA_COMP	0xE013 0820	R	PKA Comparison Result
CRYP0_PKA_DIV	0xE013 0824	R	PKA Address of Quotient MSW
CRYP0_PKA_MOD	0xE013 0828	R	PKA Address of Remainder MSW
CRYP0_INT_UNMSK	0xE015 00A0	R	Interrupt Unmasked Status
CRYP0_INT_MSK	0xE015 00A4	R/W	Interrupt Masked Status
CRYP0_INT_EN	0xE015 00A8	R/W	Interrupt Mask
CRYP0_INT_CFG	0xE015 00AC	R/W	Interrupt Configuration
CRYP0_INT_DESRD	0xE015 00B0	W	Interrupt Force Descriptor Read
CRYP0_INT_DESCT	0xE015 00B4	R/W	Interrupt Descriptor Count
CRYP0_DC_CTRL	0xE016 0080	R/W	Device Control
CRYP0_DC_DEVID	0xE016 0084	R	Device ID
CRYP0_DC_DEVINF	0xE016 0088	R	Device Information
CRYP0_DMA_USRC	0xE016 0094	R/W	DMA Source Address
CRYP0_DMA_UDST	0xE016 0098	R/W	DMA Destination Address
CRYP0_DMA_UCMD	0xE016 009C	R/W	DMA Command
CRYP0_DMA_CFG	0xE016 00D4	R/W	DMA Configuration/Status
CRYP0_PRNG_STAT	0xE017 0000	R	PRNG Status
CRYP0_PRNG_CTRL	0xE017 0004	R/W	PRNG Control
CRYP0_PRNG_SDL	0xE017 0008	W	PRNG Seed Value Low

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
CRYP0_PRNG_SDH	0xE017 000C	W	PRNG Seed Value High
CRYP0_PRNG_K0L	0xE017 0010	W	PRNG Key 0 Low
CRYP0_PRNG_K0H	0xE017 0014	W	PRNG Key 0 High
CRYP0_PRNG_K1L	0xE017 0018	W	PRNG Key 1 Low
CRYP0_PRNG_K1H	0xE017 001C	W	PRNG Key 1 High
CRYP0_PRNG_RS0	0xE017 0020	R	PRNG Result 0 (31:0)
CRYP0_PRNG_RS1	0xE017 0024	R	PRNG Result 1 (63:32)
CRYP0_PRNG_RS2	0xE017 0028	R	PRNG Result 2 (95:64)
CRYP0_PRNG_RS3	0xE017 002C	R	PRNG Result 3 (127:96)
CRYP0_PRNG_LFL	0xE017 0030	R/W	PRNG LFSR Low
CRYP0_PRNG_LFH	0xE017 0034	R/W	PRNG LFSR High
Kasumi Algorithm			
KASU0_DATAIN0, KASU0_DATAOUT0	0x0 E018 0000	W R	Data Input 0, Data Output 0
KASU0_DATAIN1, KASU0_DATAOUT1	0x0 E018 0004	W R	Data Input 1, Data Output 1
KASU0_CTRL, KASU0_STAT	0x0 E018 0008	W R	Control, Status
KASU0_MODE	0x0 E018 00C0	W	Mode input
KASU0_KEY0	0x0 E018 0010	W	Key input 31:0
KASU0_KEY1	0x0 E018 0014	W	Key input 63:32
KASU0_KEY2	0x0 E018 0018	W	Key input 95:64
KASU0_KEY3	0x0 E018 001C	W	Key input 127:96
KASU0_COUNT	0x0 E018 0020	W	Count input
KASU0_CONFIG	0x0 E018 0024	W	Configuration input
KASU0_FRESH	0x0 E018 0028	W	Fresh input
PCI to 64-bit PLB Bridge			
PCIC0_CFGADDR	0x1 EEC0 0000	R/W	PCI Space Configuration Address
PCIC0_CFGDATA	0x1 EEC0 0004	R/W	PCI Space Configuration Data
PCIC0_VENDID	offset 0x8000 0000	R/W	PCI Vendor ID
PCIC0_DEVID	offset 0x8000 0002	R/W	PCI Device ID
PCIC0_CMD	offset 0x8000 0004	R/W	PCI Command
PCIC0_STATUS	offset 0x8000 0006	R/W	PCI Status
PCIC0_REVID	offset 0x8000 0008	R/W	PCI Revision ID
PCIC0_CLS	offset 0x8000 0009	R/W	PCI Class
PCIC0_CACHELS	offset 0x8000 000C	R	PCI Cache Line Size
PCIC0_LATTIM	offset 0x8000 000D	R/W	PCI Latency Timer
PCIC0_HDTYPE	offset 0x8000 000E	R	PCI Header Type
PCIC0_BIST	offset 0x8000 000F	R	PCI Built In Self Test Control
PCIC0_PTM1BAR	offset 0x8000 0014	R/W	PCI PTM 1 BAR

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
PCIC0_PTM2BAR	offset 0x8000 0018	R/W	PCI PTM 2 BAR
PCIC0_SBSYSVID	offset 0x8000 002C	R/W	PCI Subsystem Vendor ID
PCIC0_SBSYSID	offset 0x8000 002E	R/W	PCI Subsystem ID
PCIC0_CAP	offset 0x8000 0034	R	PCI Capabilities Pointer
PCIC0_INTLN	offset 0x8000 003C	R/W	PCI Interrupt Line
PCIC0_INTPN	offset 0x8000 003D	R	PCI Interrupt Pin
PCIC0_MINGNT	offset 0x8000 003E	R	PCI Minimum Grant
PCIC0_MAXLTNCY	offset 0x8000 003F	R	PCI Maximum Latency
PCIC0_ICS	offset 0x8000 0044	R/W	PCI Interrupt Control/Status
PCIC0_ERREN	offset 0x8000 0048	R/W	Error Enable
PCIC0_ERRSTS	offset 0x8000 0049	R/W	Error Status
PCIC0_BRDGOPT1	offset 0x8000 004A	R/W	PCI Bridge Options 1
PCIC0_PLBBESR0	offset 0x8000 004C	R/W	PLB Slave Error Syndrome 0
PCIC0_PLBBESR1	offset 0x8000 0050	R/W	PLB Slave Error Syndrome 1
PCIC0_PLBBEAR	offset 0x8000 0054	R/W	PLB Slave Error Address Register
PCIC0_CAPID	offset 0x8000 0058	R	Capability Identifier
PCIC0_NEXTIPTR	offset 0x8000 0059	R	Next Item Pointer
PCIC0_PMC	offset 0x8000 005A	R	Power Management Capabilities
PCIC0_PMCSR	offset 0x8000 005C	R/W	Power Management Control Status
PCIC0_PMCSRBSE	offset 0x8000 005E	R	PMCSR PCI-to-PCI Bridge Support Extensions
PCIC0_DATA	offset 0x8000 005F	R	Data
PCIC0_BRDGOPT2	offset 0x8000 0060	R/W	PCI Bridge Options 2
PCIC0_PMSCRR	offset 0x8000 0064	R/W	Power Management State Change Request Register
PCIL0_PMM0LA	0x1 EF40 0000	R/W	PMM 0 Local Address
PCIL0_PMM0MA	0x1 EF40 0004	R/W	PMM 0 Mask/Attribute
PCIL0_PMM0PCILA	0x1 EF40 0008	R/W	PMM 0 PCI Low Address
PCIL0_PMM0PCIHA	0x1 EF40 000C	R/W	PMM 0 PCI High Address
PCIL0_PMM1LA	0x1 EF40 0010	R/W	PMM 1 Local Address
PCIL0_PMM1MA	0x1 EF40 0014	R/W	PMM 1 Mask/Attribute
PCIL0_PMM1PCILA	0x1 EF40 0018	R/W	PMM 1 PCI Low Address
PCIL0_PMM1PCIHA	0x1 EF40 001C	R/W	PMM 1 PCI High Address
PCIL0_PMM2LA	0x1 EF40 0020	R/W	PMM 2 Local Address
PCIL0_PMM2MA	0x1 EF40 0024	R/W	PMM 2 Mask/Attribute
PCIL0_PMM2PCILA	0x1 EF40 0028	R/W	PMM 2 PCI Low Address
PCIL0_PMM2PCIHA	0x1 EF40 002C	R/W	PMM 2 PCI High Address
PCIL0_PTM1MS	0x1 EF40 0030	R/W	PTM 1 Memory Size/Attribute
PCIL0_PTM1LA	0x1 EF40 0034	R/W	PTM 1 Local Address
PCIL0_PTM2MS	0x1 EF40 0038	R/W	PTM 2 Memory Size/Attribute

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
PCIL0_PTM2LA	0x1 EF40 003C	R/W	PTM 2 Local Address
General Purpose Timer			
GPT0_TBC	0x1 EF60 0000	R/W	GPT Time Base Counter
GPT0_IM	0x1 EF60 0018	R/W	GPT Interrupt Mask
GPT0_ISS	0x1 EF60 001C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x1 EF60 0020	R/W	GPT Interrupt Status (Clear bits if write 1)
GPT0_IE	0x1 EF60 0024	R/W	GPT Interrupt Enable
GPT0_COMP0	0x1 EF60 0080	R/W	GPT Compare Timer 0
GPT0_COMP1	0x1 EF60 0084	R/W	GPT Compare Timer 1
GPT0_COMP2	0x1 EF60 0088	R/W	GPT Compare Timer 2
GPT0_COMP3	0x1 EF60 008C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x1 EF60 0090	R/W	GPT Compare Timer 4
GPT0_COMP5	0x1 EF60 0094	R/W	GPT Compare Timer 5
GPT0_COMP6	0x1 EF60 0098	R/W	GPT Compare Timer 6
GPT0_MASK0	0x1 EF60 00C0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x1 EF60 00C4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x1 EF60 00C8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x1 EF60 00CC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x1 EF60 00D0	R/W	GPT Compare Mask 4
GPT0_MASK5	0x1 EF60 00D4	R/W	GPT Compare Mask 5
GPT0_MASK6	0x1 EF60 00D8	R/W	GPT Compare Mask 6
GPT0_DCT0	0x1 EF60 0110	R/W	Down Count Timer
GPT0_DCIS	0x1 EF60 011C	R/W	Down Count Timer Interrupt Status
UART Port 0			
UART0_RBR	0x1 EF60 0300	R	UART 0 Receiver Buffer Register
UART0_THR		W	UART 0 Transmitter Holding Register
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB
UART0_IER	0x1 EF60 0301	R/W	UART 0 Interrupt Enable Register
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB
UART0_IIR	0x1 EF60 0302	R	UART 0 Interrupt Identification Register
UART0_FCR	0x1 EF60 0302	W	UART 0 FIFO Control Register
UART0_LCR	0x1 EF60 0303	R/W	UART 0 Line Control Register
UART0_MCR	0x1 EF60 0304	R/W	UART 0 Modem Control Register
UART0_LSR	0x1 EF60 0305	R/W	UART 0 Line Status Register
UART0_MSR	0x1 EF60 0306	R/W	UART 0 Modem Status Register
UART0_SCR	0x1 EF60 0307	R/W	UART 0 Scratch Register

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
UART Port 1			
UART1_RBR	0x1 EF60 0400	R	UART 1 Receiver Buffer Register
UART1_THR		W	UART 1 Transmitter Holding Register
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB
UART1_IER	0x1 EF60 0401	R/W	UART 1 Interrupt Enable Register
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB
UART1_IIR	0x1 EF60 0402	R	UART 1 Interrupt Identification Register
UART1_FCR	0x1 EF60 0402	W	UART 1 FIFO Control Register
UART1_LCR	0x1 EF60 0403	R/W	UART 1 Line Control Register
UART1_MCR	0x1 EF60 0404	R/W	UART 1 Modem Control Register
UART1_LSR	0x1 EF60 0405	R/W	UART 1 Line Status Register
UART1_MSR	0x1 EF60 0406	R/W	UART 1 Modem Status Register
UART1_SCR	0x1 EF60 0407	R/W	UART 1 Scratch Register
UART Port 2			
UART2_RBR	0x1 EF60 0500	R	UART 2 Receiver Buffer Register
UART2_THR		W	UART 2 Transmitter Holding Register
UART2_DLL		R/W	UART 2 Baud-rate Divisor Latch LSB
UART2_IER	0x1 EF60 0501	R/W	UART 2 Interrupt Enable Register
UART2_DLM		R/W	UART 2 Baud-rate Divisor Latch MSB
UART2_IIR	0x1 EF60 0502	R	UART 2 Interrupt Identification Register
UART2_FCR	0x1 EF60 0502	W	UART 2 FIFO Control Register
UART2_LCR	0x1 EF60 0503	R/W	UART 2 Line Control Register
UART2_MCR	0x1 EF60 0504	R/W	UART 2 Modem Control Register
UART2_LSR	0x1 EF60 0505	R/W	UART 2 Line Status Register
UART2_MSR	0x1 EF60 0506	R/W	UART 2 Modem Status Register
UART2_SCR	0x1 EF60 0507	R/W	UART 2 Scratch Register
UART Port 3			
UART3_RBR	0x1 EF60 0600	R	UART 3 Receiver Buffer Register
UART3_THR		W	UART 3 Transmitter Holding Register
UART3_DLL		R/W	UART 3 Baud-rate Divisor Latch LSB
UART3_IER	0x1 EF60 0601	R/W	UART 3 Interrupt Enable Register
UART3_DLM		R/W	UART 3 Baud-rate Divisor Latch MSB
UART3_IIR	0x1 EF60 0602	R	UART 3 Interrupt Identification Register
UART3_FCR	0x1 EF60 0602	W	UART 3 FIFO Control Register
UART3_LCR	0x1 EF60 0603	R/W	UART 3 Line Control Register
UART3_MCR	0x1 EF60 0604	R/W	UART 3 Modem Control Register
UART3_LSR	0x1 EF60 0605	R/W	UART 3 Line Status Register
UART3_MSR	0x1 EF60 0606	R/W	UART 3 Modem Status Register

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
UART3_SCR	0x1 EF60 0607	R/W	UART 3 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x1 EF60 0700	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x1 EF60 0702	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x1 EF60 0704	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x1 EF60 0705	R/W	IIC 0 High Master Address
IIC0_CNTL	0x1 EF60 0706	R/W	IIC 0 Control
IIC0_MDCNTL	0x1 EF60 0707	R/W	IIC 0 Mode Control
IIC0_STS	0x1 EF60 0708	R/W	IIC 0 Status
IIC0_EXTSTS	0x1 EF60 0709	R/W	IIC 0 Extended Status
IIC0_LSADR	0x1 EF60 070A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x1 EF60 070B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x1 EF60 070C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x1 EF60 070D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x1 EF60 070E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x1 EF60 070F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x1 EF60 0710	R/W	IIC 0 Direct Control
IIC0_INTR	0x1 EF60 0711	R	IIC 0 Interrupt
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x1 EF60 0800	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x1 EF60 0802	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x1 EF60 0804	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x1 EF60 0805	R/W	IIC 1 High Master Address
IIC1_CNTL	0x1 EF60 0806	R/W	IIC 1 Control
IIC1_MDCNTL	0x1 EF60 0807	R/W	IIC 1 Mode Control
IIC1_STS	0x1 EF60 0808	R/W	IIC 1 Status
IIC1_EXTSTS	0x1 EF60 0809	R/W	IIC 1 Extended Status
IIC1_LSADR	0x1 EF60 080A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x1 EF60 080B	R/W	IIC 1 High Slave Address
IIC1_CLKDIV	0x1 EF60 080C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x1 EF60 080D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x1 EF60 080E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x1 EF60 080F	R/W	IIC 1 Extended Control and Slave Status
IIC1_DIRECTCNTL	0x1 EF60 0810	R/W	IIC 1 Direct Control
IIC1_INTR	0x1 EF60 0811	R	IIC 1 Interrupt
Serial Port Interface			
SPI0_MODE	0x1 EF60 0900	R/W	SPI Mode Register
SPI0_RxD	0x1 EF60 0901	R	SPI Receive Data Register

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
SPI0_TxD	0x1 EF60 0902	R/W	SPI Transmit Data Register
SPI0_CR	0x1 EF60 0903	R/W	SPI Control Register
SPI0_SR	0x1 EF60 0904	R	SPI Status Register
SPI0_CDM	0x1 EF60 0906	R/W	SPI Clock Divisor Modulus Register
OPB Arbiter 0 (attached to 64-bit PLB)			
OPBA0_PR	0x1 EF60 0A00	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x1 EF60 0A01	R/W	OPB Arbiter Control Register
General-Purpose I/O 0			
GPIO0_OR	0x1 EF60 0B00	R/W	GPIO0 Output
GPIO0_TCR	0x1 EF60 0B04	R/W	GPIO0 Three-State Control Register
GPIO0_OSRL	0x1 EF60 0B08	R/W	GPIO0 Output Select Register Low
GPIO0_OSRH	0x1 EF60 0B0C	R/W	GPIO0 Output Select Register High
GPIO0_TSRL	0x1 EF60 0B10	R/W	GPIO0 Three-State Select Register Low
GPIO0_TSRH	0x1 EF60 0B14	R/W	GPIO0 Three-State Select Register High
GPIO0_ODR	0x1 EF60 0B18	R/W	GPIO0 Open Drain Register
GPIO0_IR	0x1 EF60 0B1C	R	GPIO0 Input Register
GPIO0_RR1	0x1 EF60 0B20	R/W	GPIO0 Receive Register 1
GPIO0_RR2	0x1 EF60 0B24	R/W	GPIO0 Receive Register 2
GPIO0_RR3	0x1 EF60 0B28	R/W	GPIO0 Receive Register 3
GPIO0_ISR1L	0x1 EF60 0B30	R/W	GPIO0 Input Select Register 1 Low
GPIO0_ISR1H	0x1 EF60 0B34	R/W	GPIO0 Input Select Register 1 High
GPIO0_ISR2L	0x1 EF60 0B38	R/W	GPIO0 Input Select Register 2 Low
GPIO0_ISR2H	0x1 EF60 0B3C	R/W	GPIO0 Input Select Register 2 High
GPIO0_ISR3L	0x1 EF60 0B40	R/W	GPIO0 Input Select Register 3 Low
GPIO0_ISR3H	0x1 EF60 0B44	R/W	GPIO0 Input Select Register 3 High
General-Purpose I/O 1			
GPIO1_OR	0x1 EF60 0C00	R/W	GPIO1 Output Register
GPIO1_TCR	0x1 EF60 0C04	R/W	GPIO1 Three-State Control Register
GPIO1_OSRL	0x1 EF60 0C08	R/W	GPIO1 Output Select Register Low
GPIO1_OSRH	0x1 EF60 0C0C	R/W	GPIO1 Output Select Register High
GPIO1_TSRL	0x1 EF60 0C10	R/W	GPIO1 Three-State Select Register Low
GPIO1_TSRH	0x1 EF60 0C14	R/W	GPIO1 Three-State Select Register High
GPIO1_ODR	0x1 EF60 0C18	R/W	GPIO1 Open Drain Register
GPIO1_IR	0x1 EF60 0C1C	R	GPIO1 Input Register
GPIO1_RR1	0x1 EF60 0C20	R/W	GPIO1 Receive Register 1
GPIO1_RR2	0x1 EF60 0C24	R/W	GPIO1 Receive Register 2
GPIO1_RR3	0x1 EF60 0C28	R/W	GPIO1 Receive Register 3
GPIO1_ISR1L	0x1 EF60 0C30	R/W	GPIO1 Input Select Register 1 Low

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
GPIO1_ISR1H	0x1 EF60 0C34	R/W	GPIO1 Input Select Register 1 High
GPIO1_ISR2L	0x1 EF60 0C38	R/W	GPIO1 Input Select Register 2 Low
GPIO1_ISR2H	0x1 EF60 0C3C	R/W	GPIO1 Input Select Register 2 High
GPIO1_ISR3L	0x1 EF60 0C40	R/W	GPIO1 Input Select Register 3 Low
GPIO1_ISR3H	0x1 EF60 0C44	R/W	GPIO1 Input Select Register 3 High
Ethernet to PHY Bridge (ZMII)			
ZMII0_FER	0x1 EF60 0D00	R/W	ZMII Function Enable Register
ZMII0_SSR	0x1 EF60 0D04	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x1 EF60 0D08	R/W	ZMII SMII Status Register
Ethernet MAC 0			
EMAC0_MR0	0x1 EF60 0E00	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x1 EF60 0E04	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x1 EF60 0E08	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x1 EF60 0E0C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x1 EF60 0E10	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x1 EF60 0E14	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x1 EF60 0E18	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x1 EF60 0E1C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x1 EF60 0E20	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x1 EF60 0E24	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x1 EF60 0E28	R/W	EMAC 0 VLAN TCI Register
EMAC0_PTR	0x1 EF60 0E2C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x1 EF60 0E30	R/W	EMAC 0 Individual Address Hash Table 1
EMAC0_IAHT2	0x1 EF60 0E34	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x1 EF60 0E38	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x1 EF60 0E3C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x1 EF60 0E40	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x1 EF60 0E44	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x1 EF60 0E48	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x1 EF60 0E4C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x1 EF60 0E50	R	EMAC 0 Last Source Address Low
EMAC0_LSAL	0x1 EF60 0E54	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x1 EF60 0E58	R/W	EMAC 0 Inter-Packet Gap Value Register
EMAC0_STACR	0x1 EF60 0E5C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x1 EF60 0E60	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x1 EF60 0E64	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x1 EF60 0E68	R	EMAC 0 Number of Octets Transmitted
EMAC0_OCRX	0x1 EF60 0E6C	R	EMAC 0 Number of Octets Received

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
EMAC0_IPCR	0x1 EF60 0E70	R/W	EMAC 0 Internal PCS Configuration Register
Ethernet MAC 1			
EMAC1_MR0	0x1 EF60 0F00	R/W	EMAC 1 Mode Register 0
EMAC1_MR1	0x1 EF60 0F04	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x1 EF60 0F08	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x1 EF60 0F0C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x1 EF60 0F10	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x1 EF60 0F14	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x1 EF60 0F18	R/W	EMAC 1 Interrupt Status Enable Register
EMAC1_IAHR	0x1 EF60 0F1C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x1 EF60 0F20	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x1 EF60 0F24	R/W	EMAC 1 VLAN TPID Register
EMAC1_VTCI	0x1 EF60 0F28	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x1 EF60 0F2C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x1 EF60 0F30	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x1 EF60 0F34	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x1 EF60 0F38	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x1 EF60 0F3C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x1 EF60 0F40	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x1 EF60 0F44	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x1 EF60 0F48	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x1 EF60 0F4C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x1 EF60 0F50	R	EMAC 1 Last Source Address Low
EMAC1_LSAL	0x1 EF60 0F54	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x1 EF60 0F58	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x1 EF60 0F5C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x1 EF60 0F60	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x1 EF60 0F64	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x1 EF60 0F68	R	EMAC 1 Number of Octets Transmitted
EMAC1_OCRX	0x1 EF60 0F6C	R	EMAC 1 Number of Octets Received
EMAC1_IPCR	0x1 EF60 0F70	R/W	EMAC 1 Internal PCS Configuration Register
RGMII Registers			
RGMII0_FER	0x1 EF601000	R/W	RGMII Function Enable Register
RGMII0_SSR	0x1 EF601004	R/W	RGMII Speed Select Register
Nand Flash Controller Note: Address is an offset of the EBC address			
NDFC0_CMD	0x1 xxxx 0000	R/W	NDFC Command Register
NDFC0_ADDR	0x1 xxxx 0004	R/W	NDFC Address Register
NDFC0_DATA	0x1 xxxx 0008	R/W	NDFC Data Register

Table 3-4. PPC440EPx/GRx Memory Mapped Registers (in Address order) (continued)

Register	Address	Access	Description
NDFC0_ECC0	0x1 xxxx 0010	R	NDFC ECC Register 0
NDFC0_ECC1	0x1 xxxx 0014	R	NDFC ECC Register 1
NDFC0_ECC2	0x1 xxxx 0018	R	NDFC ECC Register 2
NDFC0_ECC3	0x1 xxxx 001C	R	NDFC ECC Register 3
NDFC0_ECC4	0x1 xxxx 0020	R	NDFC ECC Register 4
NDFC0_ECC5	0x1 xxxx 0024	R	NDFC ECC Register 5
NDFC0_ECC6	0x1 xxxx 0028	R	NDFC ECC Register 6
NDFC0_ECC7	0x1 xxxx 002C	R	NDFC ECC Register 7
NDFC0_B0CR	0x1 xxxx 0030	R/W	NDFC Bank Configuration Register 0
NDFC0_B1CR	0x1 xxxx 0034	R/W	NDFC Bank Configuration Register 1
NDFC0_B2CR	0x1 xxxx 0038	R/W	NDFC Bank Configuration Register 2
NDFC0_B3CR	0x1 xxxx 003C	R/W	NDFC Bank Configuration Register 3
NDFC0_CR	0x1 xxxx 0040	R/W	NDFC Configuration Register
NDFC0_SR	0x1 xxxx 0044	R	NDFC Status Register
NDFC0_HWCTL	0x1 xxxx 0048	R/W	NDFC Direct Hardware Control Register
NDFC0_REVID	0x1 xxxx 0050	R	NDFC Revision ID Register

3.3 Instruction Classes

PowerPC Book-E architecture defines all instructions as falling into one of the following four classes, as determined by the primary opcode (and the extended opcode, if any):

1. Defined
2. Allocated
3. Preserved
4. Reserved (illegal or nop)

Refer to the *PPC440 Processor User's Manual* for details.

4. FPU Programming Model

The programming model of the PPC440 FPU describes how the following features and operations appear to programmers (The FPU is supported by the PPC440EPx only):

- Storage addressing (including storage operands, effective address calculation, and data storage addressing modes), starting on page 139
- Floating-point exceptions, starting on page 141
- Floating-point registers, starting on page 141
- Floating-point data formats, starting on page 145
- Floating-point execution models, starting on page 151
- Floating-point instructions, starting on page 155

The Book-E Enhanced PowerPC Architecture (referred to as Book-E) specifies that the floating-point unit (FPU) implements a floating-point system as defined in ANSI/IEEE Standard 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic* (referred to as IEEE 754), but the architecture requires software support to conform fully with the standard. IEEE 754 defines certain required "operations" (addition, subtraction, and so on); the term "floating-point operation" is used to refer to one of these required operations, or to the operation performed by one of the *Multiply-Add* or *Reciprocal Estimate* instructions. All floating-point operations conform to the IEEE standard, unless software sets the IEEE Mode (NI) bit to 1 in the Floating-Point Status and Control Register (FPSCR). When FPSCR[NI] = 1, floating-point operations do not necessarily conform to the IEEE standard.

Important: Before using the FPU, it must be enabled and configured in the processor MSR and CCR0 registers. Specifically, set MSR[FP] = 1 and CCR0[DAPUIB] = 0. Also, MSR[FE0, FE1] must be set as desired. Refer to the MSR and CCR0 registers in the *PPC440 Processor User's Manual* for details.

4.1 Storage Addressing

The PPC440 FPU accesses storage in the same uniform 32-bit (4GB) effective address (EA) space as the PPC440 processor core. Effective addresses are expanded into virtual addresses and then translated to 33-bit (8GB) real addresses by the memory management unit (MMU) of the processor core.

The PPC440 FPU generates an effective address whenever it executes a *Load/Store* instruction.

4.1.1 Storage Operands

Bytes in storage are numbered consecutively starting with 0. Each number is the address of the corresponding byte.

The data storage operands accessed by the PPC440 FPU load/store instructions can be words (4 bytes, or 32 bits) or double words (8 bytes, or 64 bits). The address of a storage operand is the address of its first byte (that is, of its lowest-numbered byte). Byte ordering can be either big endian or little endian, as controlled by the endian (E) storage attribute.

Operand length is implicit for each scalar storage access instruction. The operand of such a scalar storage access instruction has a "natural" alignment boundary equal to the operand length. In other words, the "natural" address of an operand is an integral multiple of the operand length. A storage operand is said to be *aligned* if it is aligned at its natural boundary; otherwise, it is said to be *unaligned*.

Data storage operands for storage access instructions have the following characteristics.

Table 4-1. Data Operand Definitions

Storage Access Instruction Type	Operand Length	$A_{28:31}$ if aligned
Word	4 bytes	0bxx00
Doubleword	8 bytes	0bx000
Note: An "x" in an address bit position indicates that the bit can be 0 or 1 regardless of the state of other bits in the address.		

The alignment of the operand effective address of some storage access instructions can affect performance, and in some cases can cause an Alignment exception to occur. For such storage access instructions, the best performance is obtained when the storage operands are naturally aligned. Table 2-2 summarizes the effects of alignment on those storage access instruction types for which such effects exist. If an instruction type is not shown in the table, then there are no alignment effects for that instruction type.

Table 4-2. Alignment Effects for Storage Access Instructions

Storage Access Instruction Type	Alignment Effect
FP Load/Store Word	Alignment exception if the storage crosses a 16-byte boundary ($EA_{28:31}$ = 0b1100); otherwise, no effect
FP Load/Store Doubleword	Alignment exception if the storage crosses 16-byte boundary ($EA_{28:31}$ > 0b1000); otherwise no effect

Instruction storage operands, on the other hand, are a word, and the effective addresses calculated by branch instructions are therefore always word-aligned.

4.1.2 Effective Address Calculation

For a storage access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address of $2^{32} - 1$ (that is, the storage operand itself crosses the maximum address boundary), the result of the operation is undefined, as specified by the architecture. The PPC440EPx performs the operation as if the storage operand wrapped around from the maximum effective address to effective address 0. Software, however, should not depend upon this behavior, so that can be ported to other implementations that do not handle such accesses in the same manner. Software should ensure that no data storage operands cross the maximum address boundary.

Note that because instructions are words, and because the effective addresses of instructions are always implicitly on word boundaries, an instruction storage operand cannot cross any word boundary, including the maximum address boundary.

Effective address arithmetic, which calculates the starting address for storage operands, wraps around from the maximum address to address 0, for all effective address computations except next sequential instruction fetching.

4.1.3 Data Storage Addressing Modes

The PPC440 FPU supports the following data storage addressing modes.

- Base + displacement (D-mode) addressing mode:

  The 16-bit D field is sign-extended to 32 bits and added to the contents of the GPR designated by RA, or to zero if RA = 0. The low-order 32 bits of the sum form the effective address of the data storage operand.

- Base + index (X-mode) addressing mode:

The contents of the GPR designated by RB (or the value 0 for lswi and stswi) are added to the contents of the GPR designated by RA, or to zero if RA = 0; the low-order 32 bits of the sum form the effective address of the data storage operand.

4.2 Floating-Point Exceptions

Each floating-point exception, and each category of Invalid Operation Exception, is associated with an exception bit in the FPSCR. The following floating-point exceptions are detected by the processor; the associated FPSCR fields are listed with each exception and Invalid Operation exception category:

Table 4-3. Invalid Operation Exception Categories

Category	FPSCR Field
SNaN	VXSNAN
Infinity – Infinity	VXISI
Infinity ÷ Infinity	VXIDI
Zero ÷ Zero	VXZDZ
Infinity × Zero	VXIMZ
Invalid Compare	VXVC
Software Request	VXSOFT
Invalid Square Root	VXSQRT
Invalid Integer Convert	VXCVI

- Invalid Operation Exception (VX)
- Zero Divide Exception (ZX)
- Overflow Exception (OX)
- Underflow Exception (UX)
- Inexact Exception (XI)

Each floating-point exception also has a corresponding enable bit in the FPSCR. See *Floating-Point Status and Control Register Instructions* on page 161 for descriptions of these exception and enable bits, and *Floating Point Unit Interrupts and Exceptions* on page 249 for a detailed discussion of floating-point exceptions, including the effects of the FPSCR enable bits.

4.3 Floating-Point Registers

This section provides an overview of the register types implemented in the PPC440 FPU. Detailed descriptions of the floating-point registers are provided within the chapters covering the functions with which they are associated. An alphabetical summary of all registers, including bit definitions, is provided in *Register Summary* on page 947.

Certain bits in some registers are *reserved* and thus not necessarily implemented. For all registers with fields marked as reserved, these reserved fields should be written as 0 and read as *undefined*. The recommended coding practice is to perform the initial write to a register with reserved fields set to 0, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register; use logical instructions to alter defined fields, leaving reserved fields unmodified; and write the register.

Each register is classified as being of a particular *type*, as characterized by the specific instructions used to read and write registers of that type. The registers contained within the PPC440 are defined by Book-E, except for Device Control Registers (DCRs) that are implementation-specific and unique to the PPC440 FPU.

4.3.1 Register Types

The PPC440EPx floating point unit provides three types of registers, Floating Point Registers (FPRs), the FPSCR, and DCRs. Each type is characterized by the instructions used to read and write the registers. The following subsections provide an overview of each register type and the instructions associated with them.

4.3.1.1 Floating-Point Registers (FPR0:FPR31)

The PPC440 FPU provides 32 Floating-Point Registers (FPRs), each 64 bits wide. In any cycle, the FPR file can read the operands for a store instruction and an arithmetic instruction, or write the data from a load instruction and the result of an arithmetic instruction.

Figure 4-1. Floating-Point Registers (FPR0:FPR31)

0:63	FPRD	Floating-Point Register data	

The FPRs are numbered FPR0:FPR31. The floating-point instruction formats provide 5-bit fields to specify the FPRs used as operands in the execution of the associated instructions.

Each FPR contains 64 bits that support the floating-point double format. All instructions that interpret the contents of an FPR as a floating-point value uses the floating-point double format for this interpretation.

The computational instructions, and the *Move* and *Select* instructions, operate on data located in FPRs and, with the exception of the *Compare* instructions, place the result value into a FPR and optionally place status information into the Condition Register (CR).

Load and store double instructions are provided that transfer 64 bits of data between storage and the FPRs with no conversion. *Load Single* instructions transfer and convert floating-point values in floating-point single format from storage to the same value in floating-point double format in the FPRs. Store single instructions are provided to transfer and convert floating-point values in floating-point double format from the FPRs to the same value in floating-point single format in storage.

Some floating-point instructions update the FPSCR and CR explicitly. Some of these instructions move data to and from an FPR to the FPSCR, or from the FPSCR to an FPR.

The computational instructions and the *Select* instruction accept values from the FPRs in double format. For single-precision arithmetic instructions, all input values must be representable in single format;. if not, the result placed into the target FPR, and the setting of status bits in the FPSCR are undefined.

4.3.1.2 Floating-Point Status and Control Register (FPSCR)

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. See *Floating-Point Status and Control Register* on page 259 for a more detailed description of the FPSCR.

Figure 4-2. Floating-Point Status and Control Register (FPSCR)

0	FX	Floating-Point Exception Summary 0 No FPSCR exception bits changed from 0 to 1. 1 At least one FPSCR exception bit changed from 0 to 1.	All floating-point instructions, except mtfsfi and mtfsf, implicitly set this field to 1 if the instruction causes any floating-point exception bits in the FPSCR to change from 0 to 1. mcrfs, mtfsfi, mtfsf, mtfsb0, and mtfsb1 can alter this field explicitly.
1	FEX	Floating-Point Enabled Exception Summary	The OR of all the floating-point exception fields masked by their respective enable fields. mcrfs, mtfsfi, mtfsf, mtfsb0, and mtfsb1 cannot alter this field explicitly.
2	VX	Floating-Point Invalid Operation Exception Summary	The OR of all the Invalid Operation exception fields. mcrfs, mtfsfi, mtfsf, mtfsb0, and mtfsb1 cannot alter this field explicitly.
3	OX	Floating-Point Overflow Exception 0 A Floating-Point Overflow exception did not occur. 1 A Floating-Point Overflow exception occurred.	See *Overflow Exception* on page 255
4	UX	Floating-Point Underflow Exception 0 A Floating-Point Underflow exception did not occur. 1 A Floating-Point Underflow exception occurred.	See *Underflow Exception* on page 256
5	ZX	Floating-Point Zero Divide Exception 0 A Floating-Point Zero Divide exception did not occur. 1 A Floating-Point Zero Divide exception occurred.	See *Zero Divide Exception* on page 255
6	IX	Floating-Point Inexact Exception 0 A Floating-Point Inexact exception did not occur. 1 A Floating-Point Inexact exception occurred.	This field is a sticky version of FPSCR[FI] The following rules describe how a given instruction sets this field. If the instruction affects FPSCR[FI], the new value of this field is obtained by ORing the old value of this field with the new value of FPSCR[FI]. If the instruction does not affect FPSCR[FI], the value of this field is unchanged.
7	VXSNAN	Floating-Point Invalid Operation Exception (SNaN) 0 A Floating-Point Invalid Operation exception (VXSNAN) did not occur. 1 A Floating-Point Invalid Operation exception (VXSNAN) occurred.	See *Invalid Operation Exception* on page 253
8	VXISI	Floating-Point Invalid Operation Exception ($\infty - \infty$) 0 A Floating-Point Invalid Operation exception (VXISI) did not occur. 1 A Floating-Point Invalid Operation exception (VXISI) occurred.	See *Invalid Operation Exception* on page 253
9	VXIDI	Floating-Point Invalid Operation Exception ($\infty \div \infty$) 0 A Floating-Point Invalid Operation exception (VXIDI) did not occur. 1 A Floating-Point Invalid Operation exception (VXIDI) occurred.	See *Invalid Operation Exception* on page 253
10	VXZDZ	Floating-Point Invalid Operation Exception ($0 \div 0$) 0 A Floating-Point Invalid Operation exception (VXZDZ) did not occur. 1 A Floating-Point Invalid Operation exception (VXZDZ) occurred.	See *Invalid Operation Exception* on page 253

11	VXIMZ	Floating-Point Invalid Operation Exception ($\infty \times 0$) 0 A Floating-Point Invalid Operation exception (VXIMZ) did not occur. 1 A Floating-Point Invalid Operation exception (VXIMZ) occurred.	See *Invalid Operation Exception* on page 253
12	VXVC	Floating-Point Invalid Operation Exception (Invalid Compare) 0 A Floating-Point Invalid Operation exception (VXVC) did not occur. 1 A Floating-Point Invalid Operation exception (VXVC) occurred.	See *Invalid Operation Exception* on page 253
13	FR	Floating-Point Fraction Rounded	The last *Arithmetic* or *Rounding and Conversion* instruction either produced an inexact result during rounding or caused a disabled Overflow Exception. See *Rounding* on page 150. This bit is not sticky.
14	FI	Floating-Point Fraction Inexact	The last *Arithmetic* or *Rounding and Conversion* instruction either produced an inexact result during rounding or caused a disabled Overflow Exception. See *Rounding* on page 150. This bit is not sticky. See the definition of FPSCR[XX] regarding the relationship between FPSCR[FI] and FPSCR[XX].
15	FPRF	Floating-Point Result Flag (FPRF)	
16	FL	Floating-Point Less Than or Negative	
17	FG	Floating-Point Greater Than or Positive	
18	FE	Floating-Point Equal to Zero	
19	FU	Floating-Point Unordered or NaN	
20		Reserved	
21	VXSOFT	Floating-Point Invalid Operation Exception (Software Request) 0 A Floating-Point Invalid Operation exception (Software Request) did not occur. 1 A Floating-Point Invalid Operation exception (Software Request) occurred.	See *Invalid Operation Exception* on page 253
22	VXSQRT	Floating-Point Invalid Operation Exception (Invalid Square Root) 0 A Floating-Point Invalid Operation exception (Invalid Square Root) did not occur. 1 A Floating-Point Invalid Operation exception (Invalid Square Root) occurred.	See *Invalid Operation Exception* on page 253
23	VXCVI	Floating-Point Invalid Operation Exception (Invalid Integer Convert) 0 A Floating-Point Invalid Operation exception (Invalid Integer Convert) did not occur. 1 A Floating-Point Invalid Operation exception (Invalid Integer Convert) occurred.	See *Invalid Operation Exception* on page 253
24	VE	Floating-Point Invalid Operation Exception Enabled 0 Floating-Point Invalid Operation exceptions are disabled. 1 Floating-Point Invalid Operation exceptions are enabled.	
25	OE	Floating-Point Overflow Exception Enable 0 Floating-Point Overflow exceptions are disabled. 1 Floating-Point Overflow exceptions are enabled.	

26	UE	Floating-Point Underflow Exception Enable 0 Floating-Point Underflow exceptions are disabled. 1 Floating-Point Underflow exceptions are enabled.	
27	ZE	Floating-Point Zero Divide Exception Enable 0 Floating-Point Zero Divide exceptions are disabled. 1 Floating-Point Zero Divide exceptions are enabled.	
28	XE	Floating-Point Inexact Exception Enable 0 Floating-Point Inexact exceptions are disabled. 1 Floating-Point Inexact exceptions are enabled.	
29	NI	Floating-Point Non-IEEE Mode 0 Non-IEEE mode is disabled 1 Non-IEEE mode is enabled.	If FPSCR[NI = 1, the remaining FPSCR bits may have meanings other than those given in this document, and the results of floating-point operations need not conform to the IEEE standard. If the IEEE-conforming result of a floating-point operation would be a denormalized number, the result of that operation is 0 (with the same sign as the denormalized number) if FPSCR[NI] = 1. The behavior when FPSCR[NI] = 1 can vary from one implementation to another
30:31	RN	Floating-Point Rounding Control 00 Round to nearest 01 Round toward zero 10 Round toward +Infinity 11 Round toward –Infinity	See *Rounding* on page 150.

Programming Note: Setting FPSCR[NI] = 1 is intended to permit results to be approximate and to cause performance to be more predictable and less data-dependent than when FPSCR[NI] = 0. For example, in non-IEEE mode, 0 is returned instead of a denormalized number, and non-IEEE mode may return a large number instead of an infinity. In non-IEEE mode, an implementation should provide a means for ensuring that all results are produced without software assistance (that is, without causing an Enabled exception type Program interrupt or a Floating-Point Unimplemented Instruction exception type Program interrupt, and without invoking an "emulation assist." See *Floating Point Unit Interrupts and Exceptions* on page 249 The means may be controlled by one or more other FPSCR bits (recall that the other FPSCR bits have implementation-dependent meanings when FPSCR[NI] = 1).

4.4 Floating-Point Data Formats

Floating-point values are represented in two binary fixed-length formats. Single-precision values are represented in the 32-bit single format. Double-precision values are represented in the 64-bit double format. The single format can be used for data in storage, but cannot be stored in the FPRs. The double format can be used for data in storage and for data in the FPRs. When a floating-point value is loaded from storage using a *Load Single* instruction, it is converted to double format and placed in the target FPR. Conversely, a floating-point value stored from an FPR into storage using a Store Single instruction is converted to single format before being placed in storage.

The lengths of the exponent and the fraction fields differ between these two formats. The structure of the single and double formats are shown in *Table 4-4* and *Table 4-5*, respectively.

Table 4-4. Floating-Point Single Format

S	EXP	FRACTION	
0	1	9	31

Table 4-5. Floating-Point Double Format

S	EXP	FRACTION	
0	1	12	63

Values in floating-point format are composed of three fields:

Table 4-6. Format Fields

Field	Description
S	Sign Bit
EXP	Exponent + bias
FRACTION	Fraction

If only a portion of a floating-point data item in storage is accessed, such as with a load or store instruction for a byte or half word (or word in the case of floating-point double format), the value affected depends on whether the PowerPC Embedded system is operating with big endian or little endian byte ordering.

4.4.1 Value Representation

Representation of numeric values in the floating-point formats consists of a sign bit (S), a biased exponent (EXP), and the fraction portion (FRACTION) of the significand. The significand consists of a leading implied bit concatenated on the right with the FRACTION. This leading implied bit is 1 for normalized numbers and 0 for denormalized numbers and is located in the unit bit position (that is, the first bit to the left of the binary point). Values representable within the two floating-point formats can be specified by the parameters listed in *Table 4-7*.

Table 4-7. IEEE 754 Floating-Point Fields

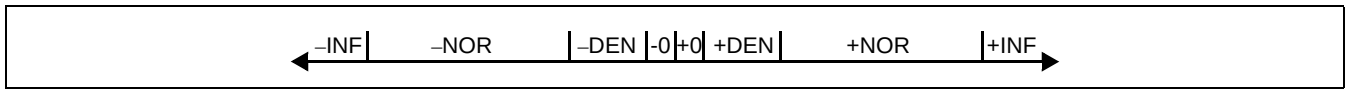
	Single	Double
Exponent Bias	+127	+1023
Maximum Exponent	+127	+1023
Minimum Exponent	–126	–1022
Field Widths (Bits)		
Sign	1	1
Exponent	8	11
Fraction	23	52
Significand	24	53

The FPRs support the floating-point double format only.

The numeric and nonnumeric values representable within each of the two supported formats are approximations to the real numbers and include the normalized numbers, denormalized numbers, and zero values. The nonnumeric values representable are the infinities and the Not a Numbers (NaNs). The infinities are adjoined to the real numbers, but are not numbers themselves, and the standard rules of arithmetic do not hold when they are used in

an operation. They are related to the real numbers by order alone. It is possible, however. to define restricted operations among numbers and infinities. The relative location on the real number line for each of the defined entities is shown in *Figure 4-3*.

Figure 4-3. Approximation to Real Numbers



The NaNs are not related to the numeric values or infinities by order or value, but are encodings used to convey diagnostic information such as the representation of uninitialized variables.

The following is a description of the different floating-point values defined in the architecture:

4.4.2 Binary Floating-Point Numbers

Machine-representable values used as approximations to real numbers. Three categories of numbers are supported: normalized numbers, denormalized numbers, and zero values.

4.4.2.1 Normalized Numbers

Normalized numbers (±NOR) have an unbiased exponent value in the range:

- –126 to 127 in single format
- –1022 to 1023 in double format

They are values in which the implied unit bit is 1. Normalized numbers are interpreted as follows:

$$NOR = (-1)^{s} \times 2^{E} \times (1.fraction)$$

where s is the sign, E is the unbiased exponent, and 1.fraction is the significand, which is composed of a leading unit bit (implied bit) and a fraction part.

The ranges covered by the magnitude (M) of a normalized floating-point number are approximately equal to:

- Single Format:

  $1.2 \times 10^{-38} \leq M \leq 3.4 \times 10^{38}$

- Double Format:

  $2.2 \times 10^{-308} \leq M \leq 1.8 \times 10^{308}$

4.4.2.2 Denormalized Numbers

Denormalized numbers (±DEN) are values that have a biased exponent value of zero and a nonzero fraction value. They are nonzero numbers smaller in magnitude than the representable normalized numbers. They are values in which the implied unit bit is 0. Denormalized numbers are interpreted as follows:

$$DEN = (-1)^{s} \times 2^{Emin} \times (0.fraction)$$

where Emin is the minimum representable exponent value (–126 for single-precision, –1022 for double-precision).

4.4.2.3 Zero Values

Zero values (±0) have a biased exponent value of zero and a fraction value of zero. Zeros can have a positive or negative sign. The sign of zero is ignored by comparison operations; comparison treats +0 as equal to –0).

4.4.3 Infinities

Infinities (±∞)are values that have the maximum biased exponent value:

- 255 in single format
- 2047 in double format

and a zero fraction value. They are used to approximate values greater in magnitude than the maximum normalized value.

Infinity arithmetic is defined as the limiting case of real arithmetic, with restricted operations defined among numbers and infinities. Infinities and the real numbers can be related by ordering in the affine sense:

$-\infty <$ every finite number $< +\infty$

Arithmetic on infinities is always exact and does not signal any exception, except when an exception occurs due to the invalid operations as described in *Invalid Operation Exception* on page 253.

4.4.3.1 Not a Numbers

Not a Numbers (NaNs) are values that have the maximum biased exponent value and a nonzero fraction value. The sign bit is ignored, that is, NaNs are neither positive nor negative. If the high-order bit of the fraction field is 0, the NaN is a Signalling NaN (SNaN); otherwise it is a Quiet NaN (QNaN).

Signaling NaNs are used to signal exceptions when they appear as operands of computational instructions.

Quiet NaNs are used to represent the results of certain invalid operations, such as invalid arithmetic operations on infinities or on NaNs, when Invalid Operation Exception is disabled (FPSCR[VE] = 0). Quiet NaNs propagate through all floating-point instructions except **fcmpo**, **frsp**, and **fctiw**. Quiet NaNs do not signal exceptions, except for ordered comparison and conversion to integer operations. Specific encodings in QNaNs can thus be preserved through a sequence of floating-point operations, and used to convey diagnostic information to help identify results from invalid operations.

When a QNaN is the result of a floating-point operation because one of the operands is a NaN or because a QNaN was generated due to a disabled Invalid Operation exception, the following rule is applied to determine the NaN with the high-order fraction bit set to 1 that is to be stored as the result.

```
if FPR(FRA) is a NaN
then FPR(FRT) ← FPR(FRA)
else if FPR(FRB) is a NaN
then if instruction is frsp
        then FPR(FRT) ← FPR(FRB)0:34 || 29 0
        else FPR(FRT) ← FPR(FRB)
else if FPR(FRC) is a NaN
        then FPR(FRT) ← FPR(FRC)
        else if generated QNaN
                then FPR(FRT) ← generated QNaN
```

If the operand specified by FRA is a NaN, that NaN is stored as the result. Otherwise, if the operand specified by FRB is a NaN (if the instruction specifies an FRB operand), that NaN is stored as the result, with the low-order 29 bits of the result set to 0 if the instruction is **frsp**. Otherwise, if the operand specified by FRC is a NaN (if the instruction specifies an FRC operand), that NaN is stored as the result. Otherwise, if a QNaN was generated due to a disabled Invalid Operation Exception, that QNaN is stored as the result. If a QNaN is to be generated as a result, the QNaN generated has a sign bit of 0, an exponent field of all 1s, and a high-order fraction bit of 1 with all other fraction bits 0. Any instruction that generates a QNaN as the result of a disabled Invalid Operation must generate this QNaN (that is, 0x7FF8000000000000).

A double-precision NaN is representable in single format if and only if the low-order 29 bits of the double-precision NaNs fraction are zero.

4.4.4 Sign of Result

The following rules govern the sign of the result of an arithmetic, rounding, or conversion operation, when the operation does not yield an exception. They apply even when the operands or results are zeros or infinities.

- The sign of the result of an add operation is the sign of the operand having the larger absolute value. If both operands have the same sign, the sign of the result of an add operation is the same as the sign of the operands. The sign of the result of the subtract operation $x - y$ is the same as the sign of the result of the add operation $x + (-y)$.

When the sum of two operands with opposite sign, or the difference of two operands with the same sign, is exactly zero, the sign of the result is positive in all rounding modes except Round toward -Infinity, in which mode the sign is negative.

- The sign of the result of a multiply or divide operation is the Exclusive OR of the signs of the operands.
- The sign of the result of a *Square Root* or **frsqrte** instruction is always positive, except that the square root of –0 is –0 and the reciprocal square root of –0 is –Infinity.
- The sign of the result of an **frsp**[**.**], or **fctiw** operation is the sign of the operand being converted.

For the *Multiply-Add* instructions, the preceding rules are applied first to the multiply operation and then to the add or subtract operation (one of the inputs to the add or subtract operation is the result of the multiply operation).

4.4.5 Normalization and Denormalization

The intermediate result of an arithmetic or **frsp** instruction may require normalization and/or denormalization. Normalization and denormalization do not affect the sign of the result.

When an arithmetic or **frsp** instruction produces an intermediate result, consisting of a sign bit, an exponent, and a nonzero significand with a 0 leading bit, it is not a normalized number and must be normalized before it is stored.

A number is normalized by shifting its significand left while decrementing its exponent by 1 for each bit shifted, until the leading significand bit becomes 1. The G bit and the R bit (see *Execution Model for IEEE Operations* on page 152) participate in the shift with zeros shifted into the Round bit. The exponent is regarded as if its range were unlimited.

After normalization, or if normalization was not required, the intermediate result may have a nonzero significand and an exponent value that is less than the minimum value that can be represented in the format specified for the result. In this case, the intermediate result is said to be “Tiny” and the stored result is determined by the rules described in *Underflow Exception* on page 256. These rules may require denormalization.

A number is denormalized by shifting its significand right while incrementing its exponent by 1 for each bit shifted, until the exponent is equal to the format's minimum value. If any significant bits are lost in this shifting process, “Loss of Accuracy” has occurred (see *Underflow Exception* on page 256) and an Underflow Exception is signaled.

4.4.6 Data Handling and Precision

Instructions are defined to move floating-point data between the FPRs and storage. For double format data, the data are not altered during the move. For single format data, a format conversion from single to double is performed when loading from storage into an FPR. A format conversion from double to single is performed when storing from an FPR to storage. The *Load/Store* instructions do not cause floating-point exceptions.

All computational, *Move*, and **fsel** instructions use the floating-point double format.

Floating-point single-precision values are obtained with the following types of instruction.

- Load Floating-Point Single

  This form of instruction accesses a single-precision operand in single format in storage, converts it to double format, and loads it into an FPR. No floating-point exceptions are caused by these instructions.

- Round to Floating-Point Single-Precision

  The **frsp** instruction rounds a double-precision operand to single-precision, checking the exponent for single-precision range and handling any exceptions according to respective enable bits, and places that operand into an FPR as a double-precision operand. For results produced by single-precision arithmetic instructions, single-precision loads, and other instances of the **frsp** instruction, this operation does not alter the value.

Programming Note: The **frsp** instruction enables value conversion from double-precision to single-precision with appropriate exception checking and rounding. This instruction should be used to convert double-precision floating-point values (produced by double-precision load and arithmetic instructions) to single-precision values before storing them into single format storage elements or using them as operands for single-precision arithmetic instructions. Values produced by single-precision load and arithmetic instructions are already single-precision values and can be stored directly into single format storage elements, or used directly as operands for single-precision arithmetic instructions, without preceding the store, or the arithmetic instruction, by an **frsp** instruction.

- Single-Precision Arithmetic Instructions

  This form of instruction takes operands from the FPRs in double format, performs the operation as if it produced an intermediate result having infinite precision and unbounded exponent range, and then coerces this intermediate result to fit in single format. Status bits in the FPSCR are set to reflect the single-precision result. The result is then converted to double format and placed into an FPR. The result lies in the range supported by the single format.

  All input values must be representable in single format. If they are not, the result placed into the target FPR, and the setting of status bits in the FPSCR, are undefined.

- Store Floating-Point Single

  This form of instruction converts a double-precision operand to single format and stores that operand into storage. No floating-point exceptions are caused by these instructions. (The value being stored is effectively assumed to be the result of an instruction of one of the preceding three types.)

When the result of a *Load Floating-Point Single*, **frsp**, or single-precision arithmetic instruction is stored in an FPR, the low-order 29 fraction bits are zero.

Programming Note: A single-precision value can be used in double-precision arithmetic operations. The reverse is true only if the double-precision value is representable in single format.

4.4.7 Rounding

Rounding applies to operations that have numeric operands (operands that are not infinities or NaNs). Rounding the intermediate result of such operations may cause an Overflow Exception, an Underflow Exception, or an Inexact Exception. The following description assumes that the operations cause no exceptions and that the result is numeric. See *Value Representation* on page 146 and *Floating Point Unit Interrupts and Exceptions* on page 249 for the cases not covered here.

Execution Model for IEEE Operations on page 152 provides a detailed explanation of rounding.

The *Arithmetic* and *Rounding and Conversion* instructions produce intermediate results that can be regarded as having infinite precision and unbounded exponent range. Such intermediate results are normalized or denormalized if required, then rounded to the target format. The final result is then placed into the target FPR in double format or in integer format, depending on the instruction.

The *Arithmetic* and *Rounding and Conversion* instructions, which round intermediate results, set FPSCR[FR, FI]. If the fraction was incremented during rounding, FPSCR[FR] = 1; otherwise, FPSCR[FR] = 0. If the rounded result is inexact, FPSCR[FI] = 1; otherwise, FPSCR[FI] = 0.

The *Estimate* instructions set FPSCR[FR, FI] to undefined values. The remaining floating-point instructions do not alter FPSCR[FR, FI].

FPSCR[RN] specifies one of four programmable rounding modes.

Let *z* be the intermediate arithmetic result or the operand of a convert operation. If *z* can be represented exactly in the target format, then the result in all rounding modes is *z* as represented in the target format. If *z* cannot be represented exactly in the target format, let *z1* and *z2* bound *z* as the next larger and next smaller numbers representable in the target format. Then, *z1* or *z2* can be used to approximate the result in the target format.

Figure 4-4 shows the relation of *z*, *z1*, and *z2* in this case. The following rules specify the rounding in the four modes. 'lsb' means 'least-significant bit'.

Figure 4-4. Selection of z1 and z2

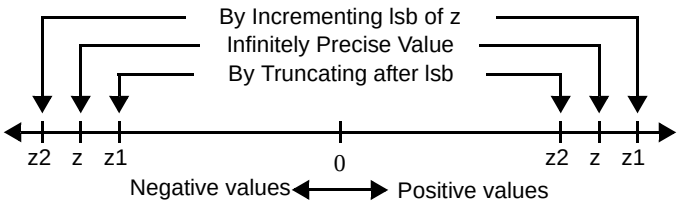


Table 4-8 describes the rounding modes.

Table 4-8. Rounding Modes

FPSCR[RN]	Rounding Mode	Description
00	Round to Nearest	Choose the value that is closest to *z*, either *z1* or *z2*. In case of a tie, choose the one that is even (the lsb is 0).
01	Round toward Zero	Choose the smaller in magnitude (*z1* or *z2*).
10	Round toward +Infinity	Choose *z1*.
11	Round toward –Infinity	Choose *z2*.

4.5 Floating-Point Execution Models

All implementations of this architecture must provide the equivalent of the following execution models to ensure that identical results are obtained.

Special rules are provided in the definition of the computational instructions for the infinities, denormalized numbers and NaNs. The material in the remainder of this section applies to instructions that have numeric operands and a numeric result (i.e., operands and result that are not infinities or NaNs), and that cause no exceptions. See *Value Representation* on page 146 and *Floating Point Unit Interrupts and Exceptions* on page 249 for the cases not covered here.

Although the double format specifies an 11-bit exponent, exponent arithmetic makes use of two additional bits to avoid potential transient overflow conditions. One extra bit is required when denormalized double-precision numbers are prenormalized. The second bit is required to permit the computation of the adjusted exponent value in the following cases when the corresponding exception enable bit is 1:

- Underflow during multiplication using a denormalized operand.
- Overflow during division using a denormalized divisor.

The IEEE standard includes 32-bit and 64-bit arithmetic. The standard requires that single-precision arithmetic be provided for single-precision operands. The standard permits double-precision floating-point operations to have either (or both) single-precision or double-precision operands, but states that single-precision floating-point operations should not accept double-precision operands. Book-E follows these guidelines: double-precision arithmetic instructions can have operands of either or both precisions, while single-precision arithmetic instructions require all operands to be single-precision. Double-precision arithmetic instructions produce double-precision values, while single-precision arithmetic instructions produce single-precision values.

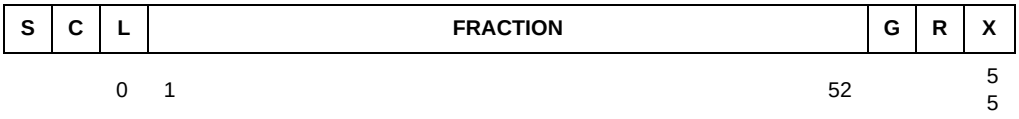
For arithmetic instructions, conversions from double-precision to single-precision must be done explicitly by software, while conversions from single-precision to double-precision are done implicitly.

4.5.1 Execution Model for IEEE Operations

The following description uses 64-bit arithmetic as an example. 32-bit arithmetic is similar except that the FRACTION is a 23-bit field, and the single-precision Guard, Round, and Sticky bits (described in this section) are logically adjacent to the 23-bit FRACTION field.

IEEE-conforming significand arithmetic is considered to be performed with a floating-point accumulator having the following format, where bits 0:55 comprise the significand of the intermediate result.

Table 4-9. IEEE 64-bit Execution Model



The S bit is the sign bit.

The C bit is the carry bit, which captures the carry out of the significand.

The L bit is the leading unit bit of the significand, which receives the implicit bit from the operand.

The FRACTION is a 52-bit field that accepts the fraction of the operand.

The Guard (G), Round (R), and Sticky (X) bits are extensions to the low-order bits of the accumulator. The G and R bits are required for post-normalization of the result. The G, R, and X bits are required during rounding to determine if the intermediate result is equally near the two nearest representable values. The X bit serves as an extension to the G and R bits by representing the logical OR of all bits that may appear to the low-order side of the R bit, due either to shifting the accumulator right or to other generation of low-order result bits. The G and R bits participate in the left shifts with zeros being shifted into the R bit. *Table 4-10* shows the significance of the G, R, and X bits with respect to the intermediate result (IR), the representable number next lower in magnitude (NL), and the representable number next higher in magnitude (NH).

Table 4-10. Interpretation of the G, R, and X Bits

G	R	X	Interpretation
0	0	0	IR is exact

Table 4-10. Interpretation of the G, R, and X Bits (continued)

G	R	X	Interpretation
0 0 0	0 1 1	1 0 1	IR closer to NL
1	0	0	IR midway between NL and NH
1 1 1	0 1 1	1 0 1	IR closer to NH

After normalization, the intermediate result is rounded, using the rounding mode specified by FPSCR[RN]. If rounding results in a carry into C, the significand is shifted right one position and the exponent incremented by one. This yields an inexact result and possibly also exponent overflow. Fraction bits to the left of the bit position used for rounding are stored into the FPR and low-order bit positions, if any, are set to zero.

Four user-selectable rounding modes are provided through FPSCR[RN] as described in *Rounding* on page 150. For rounding, the conceptual Guard, Round, and Sticky bits are defined in terms of accumulator bits. *Table 4-11* shows the positions of the Guard, Round, and Sticky bits for double-precision and single-precision floating-point numbers in the IEEE execution model.

Table 4-11. Location of the Guard, Round, and Sticky Bits in the IEEE Execution Model

Format	Guard	Round	Sticky
Double Single	G bit 24	R bit 25	X bit OR of 26:52, G, R, X

Rounding can be treated as though the significand were shifted right, if required, until the least significant bit to be retained is in the low-order bit position of the FRACTION. If any of the Guard, Round, or Sticky bits is nonzero, then the result is inexact.

Z1 and *Z2*, as defined in *Rounding* on page 150, can be used to approximate the result in the target format when one of the following rules is used.

- Round to Nearest
  - Guard bit = 0

    The result is truncated. (Result exact (GRX = 000) or closest to next lower value in magnitude (GRX = 001, 010, or 011))
  - Guard bit = 1

    Depends on Round and Sticky bits:
    - Case a

      If the Round or Sticky bit is 1 (inclusive), the result is incremented. (Result closest to next higher value in magnitude (GRX = 101, 110, or 111))
    - Case b

      If the Round and Sticky bits are 0 (result midway between closest representable values), then if the low-order bit of the result is 1 the result is incremented. Otherwise (the low-order bit of the result is 0) the result is truncated (this is the case of a tie rounded to even).
- Round toward Zero

Choose the smaller in magnitude of *Z1* or *Z2*. If the Guard, Round, or Sticky bit is nonzero, the result is inexact.

- Round toward +Infinity

  Choose *Z1*.

- Round toward –Infinity

  Choose *Z2*.

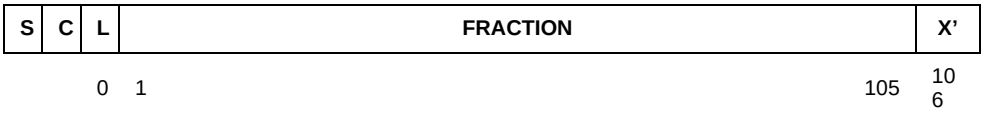
Where the result is to have fewer than 53 bits of precision because the instruction is a *Floating Round to Single-Precision* or single-precision arithmetic instruction, the intermediate result is either normalized or placed in correct denormalized form before being rounded.

4.5.2 Execution Model for Multiply-Add Type Instructions

The PPC440 FPU provides a special form of instruction that performs up to three operations in one instruction (a multiplication, an addition, and a negation). With this added capability comes the special ability to produce a more exact intermediate result as input to the rounder. 32-bit arithmetic is similar except that the FRACTION field is smaller.

Multiply-add significand arithmetic is considered to be performed with a floating-point accumulator having the following format, where bits 0:106 comprise the significand of the intermediate result.

Table 4-12. Multiply-Add 64-bit Execution Model



The first part of the operation is a multiplication. The multiplication has two 53-bit significands as inputs, which are assumed to be prenormalized, and produces a result conforming to the above model. If there is a carry out of the significand (into the C bit), then the significand is shifted right one position, shifting the L bit (leading unit bit) into the most significant bit of the FRACTION and shifting the C bit (carry out) into the L bit. All 106 bits (L bit, the FRACTION) of the product take part in the add operation. If the exponents of the two inputs to the adder are not equal, the significand of the operand with the smaller exponent is aligned (shifted) to the right by an amount that is added to that exponent to make it equal to the other input's exponent. Zeros are shifted into the left of the significand as it is aligned and bits shifted out of bit 105 of the significand are ORed into the X' bit. The add operation also produces a result conforming to the above model with the X' bit taking part in the add operation.

The result of the addition is then normalized, with all bits of the addition result, except the X' bit, participating in the shift. The normalized result serves as the intermediate result that is input to the rounder.

For rounding, the conceptual Guard, Round, and Sticky bits are defined in terms of accumulator bits. *Table 4-13* shows the positions of the Guard, Round, and Sticky bits for double-precision and single-precision floating-point numbers in the multiply-add execution model.

Table 4-13. Location of Guard, Round, and Sticky Bits in the Multiply-Add Execution Model

Format	Guard	Round	Sticky
Double	53	54	OR of 55:105, X'
Single	24	25	OR of 26:105, X'

The rules for rounding the intermediate result are the same as those given in *Execution Model for IEEE Operations* on page 152.

If the instruction is *Floating Negative Multiply-Add* or *Floating Negative Multiply-Subtract*, the final result is negated.

4.6 Floating-Point Instructions

Primary opcode 63 is used for the double-precision arithmetic instructions and miscellaneous instructions, such as the *Floating-Point Status and Control Register Manipulation* instructions. Primary opcode 59 is used for the single-precision arithmetic instructions.

The single-precision instructions for which there is a corresponding double-precision instruction have the same format and extended opcode as the corresponding double-precision instruction.

Instructions are provided to perform arithmetic, rounding, conversion, comparison, and other operations in floating-point registers; to move floating-point data between storage and these registers; and to manipulate the FPSCR explicitly.

These instructions are divided into two categories.

- Computational instructions

  The computational instructions are those that perform addition, subtraction, multiplication, division, extracting the square root, rounding, conversion, comparison, and combinations of these operations. These instructions provide the floating-point operations. They place status information into the FPSCR. They are the instructions described in *Floating-Point Arithmetic Instructions* on page 159, *Floating-Point Rounding and Conversion Instructions* on page 160, and *Floating-Point Compare Instructions* on page 160.

- Noncomputational instructions

  The noncomputational instructions that perform loads and stores, move the contents of a floating-point register to another floating-point register possibly altering the sign, manipulate the FPSCR explicitly, and select a value from one of two floating-point registers based on the value in a third floating-point register. These operations are not considered floating-point operations. With the exception of the instructions that manipulate the FPSCR explicitly, they do not alter the FPSCR. Those instructions are described in *Floating-Point Status and Control Register Instructions* on page 161.

A floating-point number consists of a signed exponent and a signed significand. The quantity expressed by this number is the product of the significand and the number $2^{exponent}$. Encodings are provided in the data format to represent finite numeric values, ±Infinity, and values that are "not a number" (NaN). Operations involving infinities produce results following traditional mathematical conventions. NaNs have no mathematical interpretation, but their encoding supports a variable diagnostic information field. NaNs may be used to indicate such things as uninitialized variables, and can be produced by certain invalid operations.

One class of exceptions that occur during floating-point instruction execution is unique to floating-point operations: the Floating-point exception. Bits set in the FPSCR indicate floating-point exceptions. They can cause an Enabled exception type Program interrupt to be taken, precisely or imprecisely, if the proper control bits are set.

4.6.1 Instructions By Category

The floating-point instructions can be classified into computational and noncomputational categories. The computational instructions include those that perform arithmetic operations or conversions on operands. Noncomputational instructions perform loads/stores and moves (with possible sign changes), or select data. Additionally, some noncomputational instructions can write directly to the FPSCR. All instructions executed in the Load/Store Pipeline are noncomputational, while most executed in the Arithmetic pipe are computational.

All floating-point operands are stored internally in Double Precision Format. Arithmetic operations specified as Single, require that the internal data is representable as Single (that is, having an unbiased exponent between –126 and 127 and a significand accurately representable in 24 bits). If the data cannot be represented in this way, the results stored in FPR, and the status bits set in FPSCR and CR (as appropriate), are undefined.

For consistency, to reduce the likelihood of causing a serious malfunction due to user error, and to enable random testing, single-precision operations are performed on double-precision operands. For all cases except for **fdivs**, the operation is performed as if it were double-precision; the result is then rounded to single-precision. For **fdivs**, the appropriate number of iterations are performed to accomplish a single-precision result (potentially with early out); the quotient is then properly rounded.

In all cases, result exceptions (overflow, underflow, and inexact) are detected and reported based on the result, not on the source operands. Default (masked exception) results are the same as for the single-precision instructions. In the case of masked overflow or underflow exceptions, the least significant 11 bits of the adjusted true exponent are returned.

The results of all single-precision operations are rounded to Single Precision. These results are stored in Double-Precision format, but are restricted to Single-Precision range (exponent and fraction). All status bits are set based upon the single-precision result.

4.6.2 Load and Store Instructions

The PPC440 FPU instruction set includes instructions to load from memory to an FPR, and to store from an FPR to memory.

For load instructions, the function of the LSC is to receive data from the 16 byte bus from the PPC440EPx and present it to the FPRs. Data received from PPC440EPx Embedded Processor could be single or double precision, and in the big or little endian formats. Also, the data received is word aligned. Data to the FPR must be in the big endian, double precision format.

For store instructions one operand from the FPR, or a bypass path, is received. Data is to be word aligned on the output bus, not all cases can be handled with a throughput of one per cycle. FPR data (or bypassed data) for single precision stores that needs to be denormalized to fit in the single precision format requires multiple cycles. Also data for double precision stores may need to be normalized if the original data source was a single precision denormalized number. There are two basic forms of load instruction: single-precision and double-precision. Because the FPRs support only floating-point double format, single-precision *Load Floating-Point* instructions convert single-precision data to double format prior to loading the operand into the target FPR. The conversion and loading steps are as follows.

Let $WORD_{0:31}$ be the floating-point single-precision operand accessed from storage.

Normalized Operand
if $WORD_{1:8} > 0$ and $WORD_{1:8} < 255$ then
$FPR(FRT)_{0:1} \leftarrow WORD_{0:1}$
$FPR(FRT)_{2} \leftarrow \neg WORD_{1}$
$FPR(FRT)_{3} \leftarrow \neg WORD_{1}$
$FPR(FRT)_{4} \leftarrow \neg WORD_{1}$
$FPR(FRT)_{5:63} \leftarrow WORD_{2:31} \,||\, {}^{29}0$

Denormalized Operand
if $WORD_{1:8} = 0$ and $WORD_{9:31} \neq 0$ then
$sign \leftarrow WORD_{0}$
$exp \leftarrow -126$
$frac_{0:52} \leftarrow 0b0 \,||\, WORD_{9:31} \,||\, {}^{29}0$
normalize the operand
do while $frac_{0} = 0$

frac ← $frac_{1:52}$ || 0b0
exp ← exp - 1
$FPR(FRT)_0$ ← sign
$FPR(FRT)_{1:11}$ ← exp + 1023
$FPR(FRT)_{12:63}$ ← $frac_{1:52}$
Zero / Infinity / NaN
if $WORD_{1:8}$ = 255 or $WORD_{1:31}$ = 0 then
$FPR(FRT)_{0:1}$ ← $WORD_{0:1}$
$FPR(FRT)_2$ ← $WORD_1$
$FPR(FRT)_3$ ← $WORD_1$
$FPR(FRT)_4$ ← $WORD_1$
$FPR(FRT)_{5:63}$ ← $WORD_{2:31}$ || $^{29}0$

For double-precision *Load Floating-Point* instructions no conversion is required, as the data from storage are copied directly into the FPR.

Some of the *Floating-Point Load* instructions update GPR(RA) the effective address. For these forms, if RA≠0, the effective address is placed into GPR(RA) and the storage element (byte, half word, word, or double word) addressed by EA is loaded into FPR(RT). If RA=0, the instruction form is invalid.

Floating-Point Load storage accesses cause Data Storage exceptions if the program is not allowed to read the storage location. *FLoating-Point Load* storage accesses cause Data TLB Error exceptions if the program attempts to access storage that is unavailable.

Note: RA and RB denote GPRs, while FRT denotes an FPR.

Both big endian and little endian byte orderings are supported.

Table 4-14. Floating-Point Load Instructions

Mnemonic	Operands	Instruction	Page
lfd	FRT, D(RA)	Load Floating-Point Double	924
lfdu	FRT, D(RA)	Load Floating-Point Double with Update	925
lfdux	FRT, RA, R	Load Floating-Point Double with Update Indexed	926
lfdx	FRT, RA, R	Load Floating-Point Double Indexed	927
lfs	FRT, D(RA)	Load Floating-Point Single	928
lfsu	FRT, D(RA)	Load Floating-Point Single with Update	929
lfsux	FRT, RA, RB	Load Floating-Point Single with Update Indexed	930
lfsx	FRT, RA, RB	Load Floating-Point Single Indexed	931

4.6.3 Floating-Point Store Instructions

There are three basic forms of store instruction: single-precision, double-precision, and integer. The integer form is provided by the **stfiwx** instruction, described on page 942. Because the FPRs support only floating-point double format for floating-point data, single-precision *Store Floating-Point* instructions convert double-precision data to single format before storing the operand in storage. The conversion steps are as follows.

Let $WORD_{0:31}$ be the word in storage written to.

No Denormalization Required (includes Zero / Infinity / NaN)
if $FPR(FRS)_{1:11} > 896$ or $FPR(FRS)_{1:63} = 0$ then
$WORD_{0:1} \leftarrow FPR(FRS)_{0:1}$
$WORD_{2:31} \leftarrow FPR(FRS)_{5:34}$
Denormalization Required
if $874 \leq FRS_{1:11} \leq 896$ then
$sign \leftarrow FPR(FRS)_0$
$exp \leftarrow FPR(FRS)_{1:11} - 1023$
$frac \leftarrow 0b1 \,||\, FPR(FRS)_{12:63}$
denormalize operand
do while $exp < -126$
$frac \leftarrow 0b0 \,||\, frac_{0:62}$
$exp \leftarrow exp + 1$
$WORD_0 \leftarrow sign$
$WORD_{1:8} \leftarrow 0x00$
$WORD_{9:31} \leftarrow frac_{1:23}$
else WORD ← undefined

Notice that if the value to be stored by a single-precision *Store Floating-Point* instruction is larger in magnitude than the maximum number representable in single format, the first case above ("No Denormalization Required") applies. The result stored in WORD is then a well-defined value, but is not numerically equal to the value in the source register. The result of a single-precision *Load Floating-Point* from WORD will not compare equal to the contents of the original source register.

For double-precision *Store Floating-Point* instructions and for the *Store Floating-Point as Integer Word* instruction no conversion is required, as the data from the FPR are copied directly into storage.

Some of the *Floating-Point Store* instructions update GPR(RA) with the effective address. For these forms, if RA≠0, the effective address is placed into GPR(RA).

Floating-Point Store storage accesses will cause a Data Storage interrupt if the program is not allowed to write to the storage location. *Integer Store* storage accesses will cause a Data TLB Error interrupt if the program attempts to access storage that is unavailable.

Note: RA and RB denote GPRs, while FRS denotes an FPR.

Both big endian and little endian byte orderings are supported.

Table 4-15. Floating-Point Store Instructions

Mnemonic	Operands	Instruction	Page
stfd	FRS, D(RA)	Store Floating-Point Double	938
stfdu	FRS, D(RA)	Store Floating-Point Double with Update	939
stfdux	FRS, RA, RB	Store Floating-Point Double with Update Indexed	940
stfdx	FRS, RA, RB	Store Floating-Point Double Indexed	941
stfiwx	FRS, RA, RB	Store Floating-Point as Integer Word Indexed	942
stfs	FRS, D(RA)	Store Floating-Point Single	943
stfsu	FRS, D(RA)	Store Floating-Point Single with Update	944
stfsux	FRS, RA, RB	Store Floating-Point Single with Update Indexed	945
stfsx	FRS, RA, RB	Store Floating-Point Single Indexed	946

4.6.4 Floating-Point Move Instructions

These instructions copy data from one floating-point register to another, altering the sign bit (bit 0) as described in the instruction descriptions in *Floating Point Instruction Set* on page 891 for **fneg**, **fabs**, and **fnabs**. These instructions treat NaNs just like any other kind of value (for example, the sign bit of an NaN may be altered by **fneg**, **fabs**, and **fnabs**). These instructions do not alter the FSPCR.

Table 4-16. Floating-Point Move Instructions

Mnemonic	Operands	Instruction	Page
fabs	FRT, FRB	Floating Absolute Value	895
fmr	FRT, FRB	Floating Move Register	906
fnabs	FRT, FRB	Floating Negative Absolute Value	911
fneg	FRT, FRB	Floating Negate	912

4.6.5 Floating-Point Arithmetic Instructions

These instructions perform elementary arithmetic operations.

Table 4-17. Floating-Point Elementary Arithmetic Instructions

Mnemonic	Operands	Instruction	Page
fadd	FRT, FRA, FRB	Floating Add	896
fadds	FRT, FRA, FRB	Floating Add Single	897
fdiv	FRT, FRA, FRB	Floating Divide	902
fdivs	FRT, FRA, FRB	Floating Divide Single	903
fmul	FRT, FRA, FRB	Floating Multiply	909
fmuls	FRT, FRA, FRB	Floating Multiply Single	910
fres	FRT, FRB	Floating Reciprocal Estimate Single	917
frsqrte	FRT, FRB	Floating Reciprocal Square Root Estimate	919
fsub	FRT, FRA, FRB	Floating Subtract	922
fsubs	FRT, FRA, FRB	Floating Subtract Single	923

4.6.5.1 Floating-Point Multiply-Add Instructions

These instructions combine a multiply and an add operation without an intermediate rounding operation. The fraction part of the intermediate product is 106 bits wide (L bit, FRACTION), and all 106 bits take part in the add/subtract portion of the instruction.

FPSCR bits are set as follows.

- Overflow, Underflow, and Inexact Exception bits, the FR and FI bits, and the FPRF field are set based on the final result of the operation, and not on the result of the multiplication.

- Invalid Operation exception bits are set as if the multiplication and the addition were performed using two separate instructions (**fmul**[**s**], followed by **fadd**[**s**] or **fsub**[**s**]. That is, multiplication of infinity by 0 or of anything by an SNaN, and addition of an SNaN, cause the corresponding exception bits to be set.

Table 4-18. Floating-Point Multiply-Add Instructions

Mnemonic	Operands	Instruction	Page
fmadd	FRT, FRA, FRB, FRC	Floating Multiply-Add	904
fmadds	FRT, FRA, FRB, FRC	Floating Multiply-Add Single	905
fmsub	FRT, FRA, FRB, FRC	Floating Multiply-Subtract	907
fmsubs	FRT, FRA, FRB, FRC	Floating Multiply-Subtract Single	908
fnmadd	FRT, FRA, FRB, FRC	Floating Negative Multiply-Add	913
fnmadds	FRT, FRA, FRB, FRC	Floating Negative Multiply-Add Single	914
fnmsub	FRT, FRA, FRB, FRC	Floating Negative Multiply-Subtract	915
fnmsubs	FRT, FRA, FRB, FRC	Floating Negative Multiply-Subtract Single	916

4.6.6 Floating-Point Rounding and Conversion Instructions

Examples of uses of these instructions to perform various conversions can be found in Appendix X, "Floating-Point Conversions", on page XREF TBD.

Table 4-19. Floating-Point Rounding and Conversion Instructions

Mnemonic	Operand	Instruction	Page
fctiw	FRT, FRB	Floating Convert To Integer Word	900
fctiwz	FRT, FRB	Floating Convert To Integer Word and round to Zero	901
frsp	FRT, FRB	Floating Round to Single-Precision	918

4.6.7 Floating-Point Compare Instructions

The floating-point *Compare* instructions compare the contents of two floating-point registers. Comparison ignores the sign of zero (+0 is treated as equal to –0). The comparison result can be ordered or unordered.

The comparison sets one bit in the designated CR field to 1 and the other three bits to 0. FPSCR[FPCC] is set in the same way.

The CR field and FPSCR[FPCC] are set as follows.

Table 4-20. Comparison Sets

Bit	Name	Description
0	FL	(FRA) < (FRB)
1	FG	(FRA) > (FRB)
2	FE	(FRA) = (FRB)
3	FU	(FRA) ? (FRB) (unordered)

Table 4-21. Floating-Point Compare and Select Instructions

Mnemonic	Operands	Instruction	Page
fcmpo	BF, FRA, FRB	Floating Compare Ordered	898
fcmpu	BF, FRA, FRB	Floating Compare Unordered	899
fsel	FRT, FRA, FRB, FRC	Floating Select	921

4.6.8 Floating-Point Status and Control Register Instructions

Every *Floating-Point Status and Control Register* instruction synchronizes the effects of all floating-point instructions executed by a given processor. Executing a *Floating-Point Status and Control Register* instruction ensures that all floating-point instructions previously initiated by the given processor have completed before the *Floating-Point Status and Control Register* instruction is initiated, and that no subsequent floating-point instructions are initiated by the given processor until the *Floating-Point Status and Control Register* instruction has completed. In particular:

- All exceptions that will be caused by the previously initiated instructions are recorded in the FPSCR before the *Floating-Point Status and Control Register* instruction is initiated.
- All invocations of the Enabled exception type Program interrupt that will be caused by the previously initiated instructions have occurred before the *Floating-Point Status and Control Register* instruction is initiated.
- No subsequent floating-point instruction that depends on or alters the settings of any FPSCR bits is initiated until the *Floating-Point Status and Control Register* instruction has completed.

Floating-Point Load and *Floating-Point Store* instructions are not affected.

Table 4-22. Floating-Point Status and Control Register Instructions

Mnemonic	Operands	Instruction	Page
mcrfs		Move To Condition Register From FPSCR	932
mffs	FRT	Move From FPSCR	933
mtfsb0	BT	Move To FPSCR Bit 0	934
mtfsb1	BT	Move To FPSCR Bit 1	935
mtfsf	FLM, FRB	Move To FPSCR Fields	936
mtfsfi	BF,U	Move To FPSCR Field Immediate	937

5. Instruction and Data Caches

The PPC440EPx/GRx Embedded Processor provides separate instruction and data cache controllers and arrays, which allow concurrent access and minimize pipeline stalls. The storage capacity of both cache arrays is 32KB. Both cache controllers have 32-byte lines, and both are highly associative, having 64-way set-associativity. The PowerPC instruction set provides a rich set of cache management instructions for software-enforced coherency. The PPC440EPx/GRx implementation also provides special debug instructions that can directly read the tag and data arrays.

Both the data and instruction caches are parity protected against soft errors. If such errors are detected, the CPU will vector to the machine check interrupt handler, where software can take appropriate action.

Refer to the *PPC440 Processor User's Manual* for details.

6. Memory Management

The 440EPx supports a uniform, 4GB effective address (EA) space, and a 8GB (33-bit) real address (RA) space. The 440EPx memory management unit (MMU) performs address translation between effective and real addresses, as well as protection functions. With appropriate system software, the MMU supports:

- Translation of effective addresses into real addresses
- Software control of the page replacement strategy
- Page-level access control for instruction and data accesses
- Page-level storage attribute control

Refer to the *PPC440 Processor User's Manual* for details.

Part III. PPC440EPx/GRx System Operations

7. Reset and Initialization

This chapter describes the initial state of the PPC440EPx/GRx after a hardware reset, and contains a description of the initialization software required to complete initialization so that the PPC440EPx/GRx can begin executing application code. Initialization of other on-chip and off-chip system components is described in the appropriate chapter.

7.1 Reset Signals

The PPC440EPx/GRx provides two reset signals, SysReset and ExtReset. SysReset is bidirectional and ExtReset is an output.

When the SysReset signal is an input (asserted by an off-chip device), such as during power on reset (POR), the chip responds by performing a system reset as described in a following section. An external assertion of SysReset is not extended by an assertion of the open drain bidirectional SysReset driver.

As an output, the PPC440EPx/GRx asserts the SysReset signal when a system reset request is detected. The SysReset open drain driver is activated and the signal is driven low for at least 16K SysClk periods. This enables the PPC440EPx/GRx to reset itself and other devices attached to the same reset network.

The ExtReset signal is used by synchronous peripheral devices served by the external bus clock, such as ROM and external masters. During chip and system resets, ExtReset is asserted until the PerClk signal is stable and all internal resets are released. Its duration in effect is the same as that of signal RstTopAllLogicRst which resets all on-chip peripherals. See *Figure 7-2*.

7.2 Reset Types

Three types of reset, each with different scope, are possible in the PPC440EPx/GRx. A CPU reset affects only the processor core and the FPU core. Chip resets affect the processor core, FPU core and all on-chip peripherals. System resets affect the processor core, FPU core, all on-chip peripherals, and any off-chip devices connected to the PPC440EPx/GRx SysReset signal. The effects of system, chip and CPU resets on the processor core and FPU core are identical. To determine which reset type occurred, the most-recent reset (MRR) field of the Debug Status Register (DBSR) can be examined.

7.2.1 CPU Reset

A CPU reset results in a reset of the processor core and FPU core. No other on-chip logic is affected. Reset_top logic, outside the processor core, detects the CPU reset request and asserts the reset input to the processor core and FPU core. The duration of a CPU reset will last 10 system clock cycles beyond the CPU core reset request, which typically lasts 4 CPU clock cycles.

7.2.2 Chip Reset

A chip reset results in the reset of the processor core, FPU core and on-chip peripherals. A chip reset request is generated by the processor core so that the processor core, FPU core and all on-chip peripherals reset for an extended period while the CPR PLL is allowed to re-lock. The duration of a chip reset will last 10 system clocks beyond the CPR PLL locking.

During chip reset, the ExtReset signal is driven low to ensure the reset of synchronous devices that use the external bus clock signal, PerClk.

7.2.3 System Reset

A system reset results in a reset of all PPC440EPx/GRx logic, including the processor core, FPU core, internal phase-locked loops (CPR PLL), and on-chip peripherals. A system reset can be initiated externally or internally. External system resets are initiated by the assertion of the SysReset signal for at least 16 SysClk cycles. Internal system resets are initiated by either the processor core the duration of a system reset will last 10 system clocks beyond the CPR PLL locking.

When a system reset is requested internally, the bidirectional open drain SysReset signal is asserted to enable other chips to be reset at the same time. In this case, the SysReset signal is driven low for 16384 SysClk cycles, resulting in a System reset of the PPC440EPx/GRx chip and all other devices attached to the reset network connected to SysReset. During this time all internal clocks and PerClk clock toggle.

After the SysReset signal is de asserted, the CPR PLL begins its locking process, which requires about 8350 SysClk cycles when CPU feedback is selected and 16700 SysClk cycles when remote PerClk feedback is selected. During this time all internal clocks and PerClk clock toggle.

When the PLL locking process is complete, internal resets are released by Reset_top logic, and the processor core begins its initial instruction fetch.

During system reset, the ExtReset signal is driven low to ensure the reset of synchronous devices that use the external bus clock signal, PerClk.

Figure 7-1. PPC440EPx/GRx Power-on Reset Process

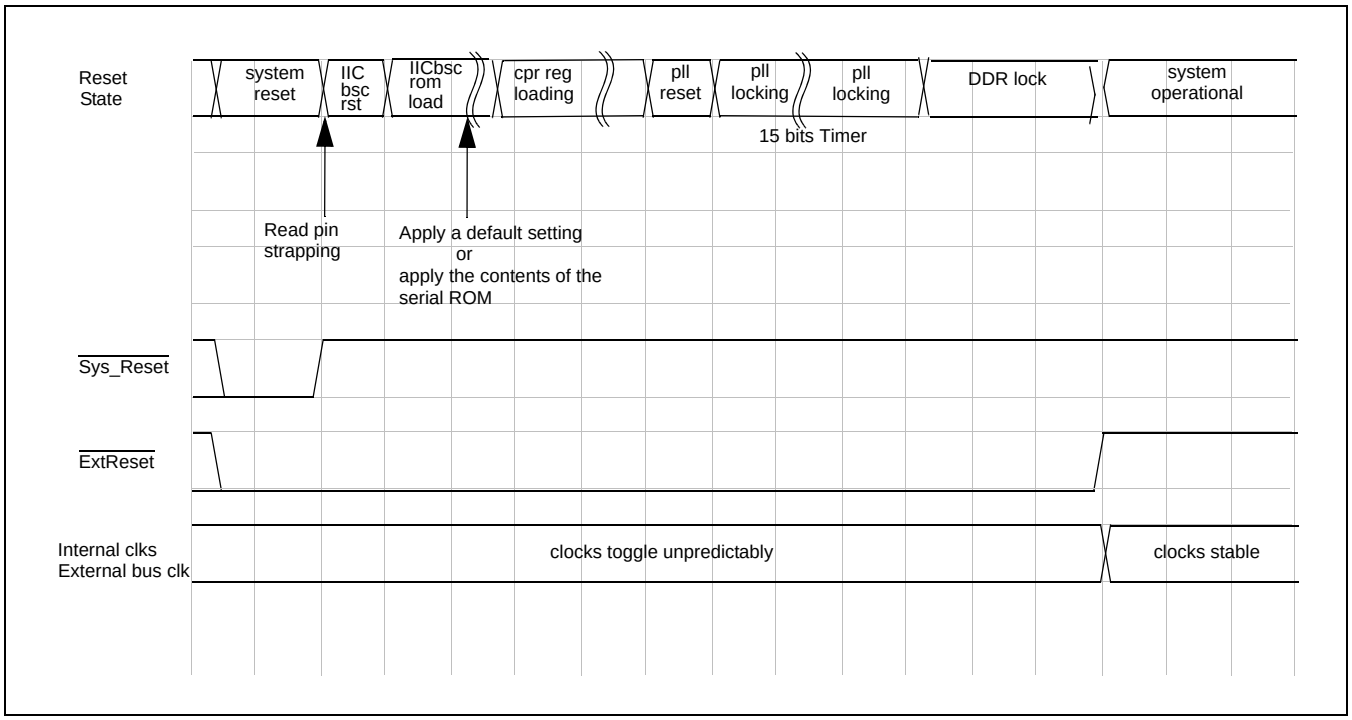
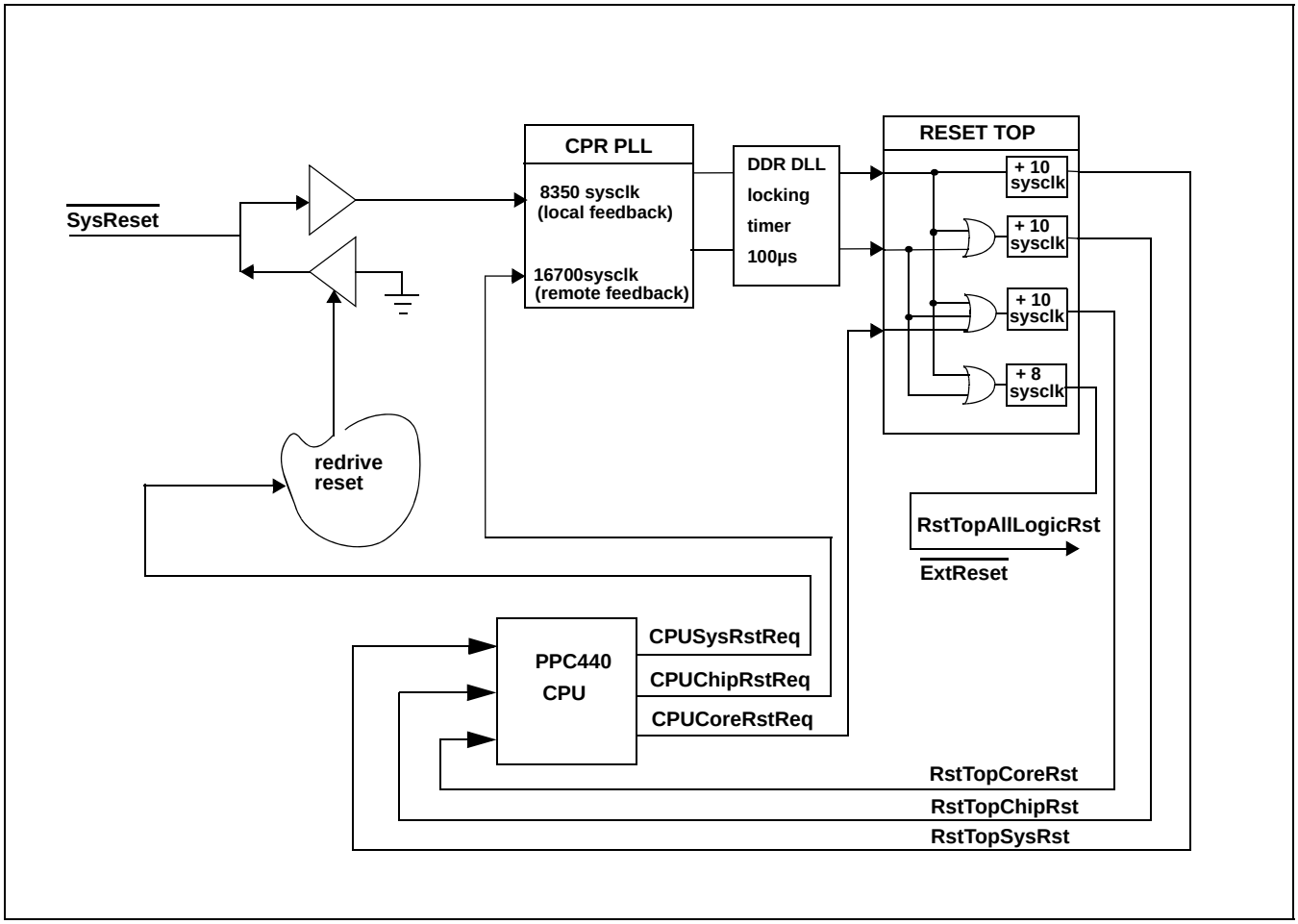


Figure 7-2. PPC440EPx/GRx Reset



7.3 Processor Core State After Reset

In general, the contents of registers and other facilities within the PPC440EPx/GRx processor core are undefined after a hardware reset. Reset is defined to initialize only the minimal resources required such that instructions can be fetched and executed from the initial program memory page, and so that repeatable, deterministic behavior can be guaranteed provided that the proper software initialization sequence is followed. System software must fully configure the rest of the PPC440EPx/GRx resources, as well as the other facilities within the chip and/or system.

The following list summarizes the requirements of the Book-E Enhanced PowerPC Architecture with regards to the processor state after reset, prior to any additional initialization by software.

- All fields of the MSR are set to 0, disabling all asynchronous interrupts, placing the processor in supervisor mode, and specifying that instruction and data accesses are to the system (as opposed to application) address space.
- DBCR0[RST] is set to 0, thereby ending any previous software-initiated reset operation.
- DBSR[MRR] records the type of the just ended reset operation (core, chip, or system; see *Reset Types* on page 169).
- TCR[WRC] is set to 0, thereby disabling the Watchdog timer reset operation.

- TSR[WRS] records the type of the just ended reset operation, if the reset was initiated by the Watchdog Timer (otherwise this field is unchanged from its pre-reset value).
- The PVR is defined, after reset and otherwise, to contain a value that indicates the specific processor implementation.
- The program counter (PC) is set to 0xFFFFFFFC, the effective address (EA) of the last word of the address space.

The memory management resources are set to values such that the processor is able to successfully fetch and execute instructions and read (but not write) data within the 4KB program memory page located at the end of the 32-bit effective address space. Exactly how this is accomplished is implementation-dependent. For example, it may or may not be the case that a TLB entry is established in a manner which is visible to software using the TLB management instructions. Regardless of how the implementation enables access to the initial program memory page, instruction execution starts at the effective address of 0xFFFFFFFC, the last word of the effective address space. The instruction at this address must be an unconditional branch backwards to the start of the initialization sequence, which must lie somewhere within the initial 4KB program memory page. The real address to which the initial effective address will be translated is also implementation- or system-dependent, as are the various storage attributes of the initial program memory page such as the caching inhibited and endian attributes.

Note: In the PPC440EPx/GRx, a single entry is established in the instruction shadow TLB (ITLB) and data shadow TLB (DTLB) at reset with the properties described in *Table 7-1 Reset Values of Registers and Other PPC440 Core Facilities* on page 172. It is required that initialization software insert an entry into the UTLB to cover this same memory region before performing any context synchronizing operation (including causing any exceptions which would lead to an interrupt), since a context synchronizing operation will invalidate the shadow TLB entries.

Initialization software should consider all other resources within the PPC440EPx/GRx to be undefined after reset, in order for the initialization sequence to be compatible with other PowerPC implementations. There are, however, additional core resources which are initialized by reset, in order to guarantee correct and deterministic operation of the processor during the initialization sequence. *Table 7-1* shows the reset state of all PPC440EPx/GRx resources which are defined to be initialized by reset. While certain other register fields and other facilities within the PPC440EPx/GRx may be affected by reset, this is not an architectural nor hardware requirement, and software must treat those resources as undefined. Likewise, even those resources which are included in *Table 7-1* but which are not identified in the previous list as being architecturally required, should be treated as undefined by the initialization software.

Table 7-1. Reset Values of Registers and Other PPC440 Core Facilities

Resource	Field	Reset Value	Comment
CCR0	DAPUIB	0	Enable broadcast of instruction data to auxiliary processor interface or FPU
	DTB	0	Enable broadcast of trace information
CCR1	ICDPEI	0	Disable Parity Error Insertion (enabled only for s/w testing)
	ICTPEI	0	
	DCTPEI	0	
	DCDPEI	0	
	DCUPEI	0	
	DCMPEI	0	
	FCOM	0	Do not force cache operations to miss.
	MMUPEI	0	Disable Parity Error Insertion (enabled only for s/w testing)
	FFF	0	Flush only as much data from dirty lines as needed.

Table 7-1. Reset Values of Registers and Other PPC440 Core Facilities (continued)

Resource	Field	Reset Value	Comment
DBCR0	EDM	0	External Debug mode disabled
	RST	0b00	Software-initiated debug reset disabled
	ICMP	0	Instruction completion debug events disabled
	BRT	0	Branch taken debug events disabled
	IAC1	0	Instruction Address Compare 1 (IAC1) debug events disabled
	IAC2	0	IAC2 debug events disabled
	IAC3	0	IAC3 debug events disabled
	IAC4	0	IAC4 debug events disabled
DBSR	UDE	0	Unconditional debug event has not occurred
	MRR	Reset dependent	Indicates most recent type of reset as follows: 00 No reset has occurred since this field last cleared by software 01 Core reset 10 Chip reset 11 System reset
	ICMP	0	Instruction completion debug event has not occurred
	BRT	0	Branch taken debug event has not occurred
	IRPT	0	Interrupt debug event has not occurred
	TRAP	0	Trap debug event has not occurred
	IAC1	0	IAC1 debug event has not occurred
	IAC2	0	IAC2 debug event has not occurred
	IAC3	0	IAC3 debug event has not occurred
	IAC4	0	IAC4 debug event has not occurred
	DAC1R	0	Data address compare 1 (DAC1) read debug event has not occurred
	DAC1W	0	DAC1 write debug event has not occurred
	DAC2R	0	DAC2 read debug event has not occurred
	DAC2W	0	DAC2 write debug event has not occurred
	RET	0	Return debug event has not occurred
ESR	MCI	0	Synchronous Instruction Machine Check exception has not occurred
MCSR	MCS	0	Asynchronous Instruction Machine Check exception has not occurred

Table 7-1. Reset Values of Registers and Other PPC440 Core Facilities (continued)

Resource	Field	Reset Value	Comment
MSR	WE	0	Wait state disabled
	CE	0	Asynchronous critical interrupts disabled
	EE	0	Asynchronous non-critical interrupts disabled
	PR	0	Processor in supervisor mode
	FP	0	Floating-point Unavailable interrupts disabled
	ME	0	Machine Check interrupts disabled
	FE0	0	Floating-point Enabled interrupts disabled
	DWE	0	Debug Wait mode disabled
	DE	0	Debug interrupts disabled
	FE1	0	Floating-point Enabled interrupts disabled
	IS	0	Instruction fetch access is to system-level virtual address space
	DS	0	Data access is to system level virtual address space
PC		0xFFFFFFFC	Initial reset instruction fetched from last word of effective address space
PVR	0:31	System dependent	Refer to *Data Sheet* for the PVR value
RSTCFG	U0	0	The U0 storage attribute has no effect in the PPC440EPx/GRx.
	U1	0	Memory page contain normal instructions or data
	U2	0	Storage misses do not cause a line to allocated in the data cache
	U3	0	The U3 storage attribute has no effect in the PPC440EPx/GRx.
	E	0	Memory pages are big endian
	ERPN	0	This value is hard wired to zero.
TCR	WRC	0b00	Watchdog Timer reset disabled

Table 7-1. Reset Values of Registers and Other PPC440 Core Facilities (continued)

Resource	Field	Reset Value	Comment
TLBentry[1]	$EPN_{0:19}$	0xFFFFF000	Match EA of initial reset instruction ($EPN_{20:21}$ are undefined, as they are not compared to the EA because the page size is 4KB).
	V	1	Translation table entry for the initial program memory page is valid.
	TS	0	Initial program memory page is in system-level virtual address space.
	SIZE	0b0001	Initial program memory page size is 4KB.
	TID	0x00	Initial program memory page is globally shared; no match required against PID register.
	$RPN_{0:21}$	0xFFFFF || 0b00	Initial program memory page mapped effective = real.
	ERPN	0b0000	Extended real page number of the initial program memory page is set to 0b0000. Shadow TLB entry at reset is 0x0FFFFF000. The reset vector is 0x0FFFFFFFC for all boot configurations.
	U0	0	The U0 storage attribute has no effect in the PPC440EPx/GRx.
	U1	0	Memory page contains normal instructions or data.
	U2	0	Storage misses do not cause a line to be allocated in the data cache.
	U3	0	The U3 storage attribute has no effect in the PPC440EPx/GRx.
	W	0	Write-through storage attribute disabled.
	I	1	Caching inhibited storage attribute enabled.
	M	0	Memory coherent storage attribute disabled.
	G	1	Guarded storage attribute enabled.
	E	0	Reset value of endian storage attribute is specified by a core input signal.
	SX	1	Supervisor mode execution access enabled.
	SW	0	Supervisor mode write access disabled.
	SR	1	Supervisor mode read access enabled.
TSR	WRS	Copy of TCR[WRC]	If reset caused by Watchdog Timer
		Unchanged	If reset not caused by Watchdog Timer
		Undefined	After power-up

Note: "TLBentry" refers to an entry in the shadow instruction and data TLB arrays that is automatically configured by the PPC440EPx/GRx to enable fetching and reading (but not writing) from the initial program memory page. This entry is not architecturally visible to software, and is invalidated upon any context synchronizing operation. Software must initialize a corresponding entry in the main unified TLB array before executing any operation which could lead to a context synchronization.

7.4 PPC440EPx/GRx Chip Initialization

The PPC440EPx/GRx chip configuration registers fall under two categories:

1. Clocking and power-on reset (CPR0) registers which are accessed indirectly through the CPR0_CFGADDR and CPR0_CFGDATA registers using the mtdcr and mfdcr instructions.
2. System device registers (SDR0) which are accessed indirectly through the SDR0_CFGADDR and SDR0_CFGDATA registers using the **mtdcr** and **mfdcr** instructions.

During PPC440EPx/GRx initialization, some chip control registers must be initialized to ensure proper chip operation. Peripheral devices are also initialized as appropriate for the system design. Registers that control PPC440EPx/GRx clocking and power-on reset are described in *Clocking* on page 273 while registers that control PPC440EPx/GRx pin strapping are described in *Bootstrap Controller* on page 187. Initialization for peripheral devices is described in individual chapters as needed including chip level serial device control registers. The section that follows, lists registers that control miscellaneous PPC440EPx/GRx devices that are not discussed in any other chapter.

7.4.1 Initial Configuration Register (CPR0_ICFG)

Figure 7-3. Initial Configuration Register (CPR0_ICFG)

0	RLI	Reload Inhibit 0 Reset CPR registers from configuration source defined by CPR0_ICFG[ICS] 1 Ignore CPR0_ICFG[ICS] configuration source and preserve current values in CPR0 registers (including CPR0_ICFG)	CPR0_ICFG[RLI] is used to preserve contents of CPR0 registers during a chip reset.
1:28		Reserved	
29:31	ICS	Initial values of the strap pins (UART0_DCD, UART0_DSR, UART0_CTS)	

7.4.2 System Device Control Registers

All SDR registers are accessed with the move-to-DCR (**mtdcr**) and move-from-DCR (**mfdcr**) instructions using indirect addressing. In indirect addressing, the **mtdcr** and **mfdcr** instructions access a given SDR register by means of its address offset which is contained in the SDR0_CFGADDR register. The data for the specified register is moved to or moved from the SDR0_CFGDATA register as described below. The SDR configuration registers are listed in *Table 7-2*. The offsets for all indirectly addressed DCR registers appear in *Table 3-3* on page 115.

Table 7-2. SDR Configuration Registers

Mnemonic	Register	Address	Access
SDR0_CFGADDR	SDR Configuration Address Register	0x000E	R/W
SDR0_CFGDATA	SDR Configuration Data Register	0x000F	R/W

Figure 7-4. SDR Configuration Address Register (SDR0_CFGADDR)

0:16			
17:31	DCRA	15-bit SDR Register Offset Value	This value can range from 0x0000 to 0x7FFF. Its contents are used as the indirect DCR address for accessing an SDR register.

Figure 7-5. SDR Configuration Data Register (SDR0_CFGDATA)			
0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contains the value of the SDR register as indicated by the SDR0_CFGADDR register.

7.4.2.1 Electronic Chip ID Register 0 (SDR0_ECID0)

The SDR0_ECIDx registers are for manufacturing use only. They uniquely identify each individual module manufactured using a 112-bit blown-fuse register.

Figure 7-6. Electronic Chip ID Register 0 (SDR0_ECID0)			
0:31	ECID0	Electronic chip ID 0	Bits 0:31 of 112-bit ID

7.4.2.2 Electronic Chip ID Register 1 (SDR0_ECID1)

Figure 7-7. Electronic Chip ID Register 1 (SDR0_ECID1)			
0:31	ECID1	Electronic chip ID 1	Bits 32:63 of 112-bit ID

7.4.2.3 Electronic Chip ID Register 2 (SDR0_ECID2)

Figure 7-8. Electronic Chip ID Register 1 (SDR0_ECID2)			
0:31	ECID2	Electronic chip ID 2	Bits 64:95 of 112-bit ID

7.4.2.4 Electronic Chip ID Register 3 (SDR0_ECID3)

Figure 7-9. Electronic Chip ID Register 3 (SDR0_ECID3)			
0:15	ECID3	Electronic chip ID 3	Bits 96:111 of 112-bit ID
16:31		Reserved	

7.4.2.5 Pin Function Control Register 0 (SDR0_PFC0)

Figure 7-10. Pin Function Control Register 0 (SDR0_PFC0)			
0:22		Reserved	Bits 0:17 reset value = 0 Bits 18:22 reset value =1
23	TRE	GPIO/Trace Enable Controls the function of the multiplexed GPIO/ Trace signals. 0 GPIO49-63 are enabled 1 Trace signals are enabled	Reset value = SDR0_SDSTP1[CTE]
24:31		Reserved	Reset value = 0

7.4.2.6 Pin Function Control Register 1 (SDR0_PFC1)

Reset value = 0x00C0100F.

Figure 7-11. Pin Function Control Register 1 (SDR0_PFC1)			
0:5		Reserved	
6	U1ME	UART1 Mode Enable Selects the function of UART1 signals in 4-pin mode. 0 Enable UART1DSR/CTS as DSR and UART1DTR/RTS as DTR 1 Enable UART1DSR/CTS as CTS and UART1DTR/RTS as RTS	See multiplexed signals GPIO34 and GPIO35 in the PPC440EPx and PPC440GRx data sheets. Note: U0IM must be set to 1 for 4-pin mode. See *Table 31-12 Alternate 2 Configuration for GPIO32–GPIO47* on page 807.
7:9	SELECT	Ethernet Pin Select EMAC0	See *Table 26-1 Ethernet Combinations* on page 681
10:11		Reserved	
12	U0ME	UART0 Mode Enable Selects the function of UART0 signals in 4-pin mode. 0 Enable UART0DSR/CTS as DSR and UART0DTR/RTS as DTR 1 Enable UART0DSR/CTS as CTS and UART0DTR/RTS as RTS	See multiplexed signals GPIO36 and GPIO37 in the PPC440EPx and PPC440GRx data sheets. Note: U0ME must be set to 1 when U0IM = 0. See *Table 31-8 Alternate 1 Configuration for GPIO32–GPIO47* on page 805.
13	U0IM	UART0 Interface Mode 0 UART0 interface mode has 8 pins 1 UART0 interface mode has 4 pins	
14	SIS	SPI/IIC1 Selection 0 SPI enabled 1 IIC1 enabled	SIS controls the function of signals: [IIC1SClk]SCPClkOut [IIC1SData]SCPDI
15	UES	USB/EBC Selection 0 USB2D enabled 1 EBC enabled	UES controls the function of signals: [HoldPri]USB2LS1 [HoldReq]USB2RxAct (The USB signals are supported by the PPC440EPx only. Set UES to 1 for the PPC440GRx).
16	DUS	DMA/UIC Selection 0 DMA enabled 1 UIC enabled	DUS controls the function of signal: [DMAReq1]IRQ5

17	ERE	External Request Enable 0 ExtReq disabled 1 ExtReq enabled	To use the EBMI, set ERE to 1. To configure GPIO27 as ExtReq see *Table 31-7 Alternate 1 Configuration for GPIO16–GPIO31* on page 804.
18:31		Reserved	

7.4.2.7 GPIO Multiplexing Register (SDR0_PFC4)

Figure 7-12. GPIO Multiplexing Register (SDR0_PFC4)

0	MUXIIC0	Select IIC0 Data or GPIO26 0 GPIO26 1 IIC0 Data	Reset value = 1
1:31		Reserved	

7.4.2.8 Slave Address Pipeline Register (SDR0_SLPIPE0)

Figure 7-13. Slave Address Pipeline Register (SDR0_SLPIPE0)

0		Reserved	
1	DAP	DDR Address Pipeline 0 DDR address pipeline disabled 1 DDR address pipeline enabled	
2	P4P3AP	PLB4 to PLB3 Address Pipeline 0 PLB4 to PLB3 address pipeline disabled 1 PLB4 to PLB3 address pipeline enabled	
3	P4OAP	PLB4 to OPB Address Pipeline 0 PLB4 to OPB address pipeline disabled 1 PLB4 to OPB address pipeline enabled	
4:27		Reserved	
28	P3OAP	PLB3 to OPB Address Pipeline 0 PLB3 to OPB address pipeline disabled 1 PLB3 to OPB address pipeline enabled	
29	EAP	EBC Address Pipeline 0 EBC address pipeline disabled 1 EBC address pipeline enabled	
30	PAP	PCI Address Pipeline 0 PCI address pipeline disabled 1 PCI address pipeline enabled	
31	P3P4AP	PLB3 to PLB4 Address Pipeline 0 PL3 to PLB4 address pipeline disabled 1 PLB3 to PLB4 address pipeline enabled	

7.4.2.9 Soft Reset Registers (SDR0_SRST0 and SDR0_SRST1)

Figure 7-14 and *Figure 7-15* describes the SDR0_SRST0 and SDR0_SRST1 register bits. For all bits, 1 = reset core, 0 = operational.

Reset value = 0 except where noted in *Figure 7-14*.

Figure 7-14. Soft Reset Register 0 (SDR0_SRST0)			
0	BGO	PLB3 to OPB bridge	
1	PLB4	PLB4 arbiter	
2	EBC	External bus controller	
3	OPB	OPB0 arbiter	
4	UART0	Universal asynchronous receiver/transmitter 0	
5	UART1	Universal asynchronous receiver/transmitter 1	
6	IIC0	Inter integrated circuit 0	
7	USB2H	USB2.0 Host *(PPC440EPx only)*	Reset = 1.
8	GPIO	General purpose I/O	
9	GPT	General purpose timer	
10	DMC	DDR SDRAM memory controller	
11	PCI	PCI	
12	EMAC0	Ethernet media access controller 0	
13	EMAC1	Ethernet media access controller 1	
14	CPM0	Clock and power management	
15	ZMII	ZMII bridge	
16	UIC0	Universal interrupt controller 0	
17	UIC1	Universal interrupt controller 1	
18	IIC1	Inter integrated circuit 1	
19	SCP	Serial communications port	
20	BGI	OPB to PLB bridge	
21	DMA	Direct memory access controller	
22	DMAC	DMA channel	
23	MAL	Media access layer	
24	USB2D	USB2.0 device *(PPC440EPx only)*	Reset = 1.
25	GPTR	General purpose timer	
26		Reserved	
27	P4P3	PLB4 to PLB3 bridge	
28	P3P4	PLB3 to PLB4 bridge	
29	PLB3	PLB3 arbiter	
30	UART2	Universal asynchronous receiver/transmitter 2	
31	UART3	Universal asynchronous receiver/transmitter 3	

Figure 7-15. Soft Reset Register 1(SDR0_SRST1)			
0	NDFC	Nand flash controller	
1	OPB1	OPB1 Arbiter attached to PLB4 *(PPC440EPx only)*	Reset = 1.
2	P4OPB0	PLB4 to OPB *(PPC440EPx only)*	Reset = 1.

3	DMA4	DMA to PLB4	
4	DMA4CH	DMA Channel to PLB4	
5	OPB2	OPB2 Arbiter attached to PLB4 USB 2.0 Host *(PPC440EPx only)*	Reset = 1.
6	PLB42OPB1	PLB4 to OPB2 Bridge attached to USB 2.0 Host *(PPC440EPx only)*	Reset = 1.
7	OPB2PLB40	OPB2 to PLB4 Bridge attached to USB 2.0 Host *(PPC440EPx only)*	Reset = 1.
8	CPM1	Clock and Power management 1	
9	UIC2	Universal Interrupt Controller 2	
10	CRYP0	Security Engine	Reset = 1.
11	USB20PHY	USB 2.0 Phy *(PPC440EPx Only)*	Reset = 1.
12	USB2HUTMI	USB 2.0 Host UTMI Interface *(PPC440EPx only)*	Reset = 1.
13	USB2HPHY	USB 2.0 Host Phy Interface *(PPC440EPx only)*	Reset = 1.
14	SRAM0	Internal SRAM Controller	
15	RGMII0	RGMII Bridge	
16	ETHPLL	Ethernet PLL	
17	FPU	Floating Point Unit *(PPC440EPx only)*	Reset = 1.
18	KASU0	Kasumi Engine	Reset = 1.
19:31		Reserved	

7.4.2.10 Miscellaneous Function Register (SDR0_MFR)

Reset value = 0 for bits 4:5 and 8:31.

Figure 7-16. Miscellaneous Function Register (SDR0_MFR)

0:3		Reserved	Reset Value = 0b0001
4	E0CS	Ethernet 0 Clock Selection 0 Ethernet 0 selects external clock 1 Ethernet 0 selects internal reference clock (for internal loop back test mode)	SDR0_MFR[E0CS] is used in conjunction with EMAC0_MR1[ILE] Note: SDR0_MFR[E0CS] will change the clock even if it is not in internal loop back mode.
5	E1CS	Ethernet 1 Clock Selection 0 Ethernet 1 selects external clock 1 Ethernet 1 selects internal reference clock (for internal loop back test mode)	SDR0_MFR[E1CS] is used in conjunction with EMAC1_MR1[ILE] Note: SDR0_MFR[E1CS] will change the clock even if it is not in internal loop back mode.
6:7	ZMII	ZMII Mode 00 MII 01 SMII 10 Reserved 11 Reserved	Reset value = SDR0_SDSTP1[ZM]
8:9		Reserved	
10	PRE0	Packet Reject for EMAC0 Enable 0 Disable 1 Enable	
11	PRE1	Packet Reject for EMAC1 Enable 0 Disable 1 Enable	

12	PRP	Packet Reject Polarity 0 Packet reject signal is active low 1 Packet reject signal is active high	
13:31		Reserved	

7.4.2.11 DDR Configuration Register (SDR0_DDRCFG)

Figure 7-17. DDR Configuration Register (SDR0_DDRCFG)

0:22		Reserved	
23	LT2	Leakage Test 0 Functional (default) 1 Manufacturing test only	SYS Must be set to 0 for normal operation.
24	64B32B	64/32 Bit Mode 0 32 bit data interface (40 with ECC), required for DDR1 1 64 bit data interface (72 with ECC), prohibited for DDR1	SYS
25	MC2518	DDR I/O Voltage Level 0 2.5V for DDR1 1 1.8V for DDR2 (default)	
26:31		Reserved	

7.5 Initialization Software Requirements

After a reset operation occurs, the PPC440EPx/GRx is initialized to a minimum configuration to enable the fetching and execution of the software initialization code, and to guarantee deterministic behavior of the core during the execution of this code. Initialization software is necessary to complete the configuration of the processor core and the rest of the on-chip and off-chip system.

The system must provide non-volatile memory (or memory initialized by some mechanism other than the PPC440EPx/GRx) at the real address corresponding to effective address 0xFFFFFFFC, and at the rest of the initial program memory page. The instruction at the initial address must be an unconditional branch backwards to the beginning of the initialization software sequence.

The initialization software functions described in this section perform the configuration tasks required to prepare the PPC440EPx/GRx to boot an operating system and subsequently execute an application program.

The initialization software must also perform functions associated with hardware resources that are outside the PPC440EPx/GRx

Initialization software should perform the following tasks in order to fully configure the PPC440EPx/GRx. For more information on the various functions referenced in the initialization sequence, see the corresponding chapters of this document.

1. Branch backwards from effective address 0xFFFFFFFC to the start of the initialization sequence
2. Invalidate the instruction cache (**iccci**)
3. Invalidate the data cache (**dccci**)
4. Synchronize memory accesses (**msync**)

This step forces any data PLB operations that may have been in progress prior to the reset operation to complete, thereby allowing subsequent data accesses to be initiated and completed properly.

5. Clear DBCR0 register (disable all debug events)

   Although the PPC440EPx/GRx is defined to reset some of the debug event enables during the reset operation (as specified in *Table 7-1* on page 172), this is not required by the architecture and hence the initialization software should not assume this behavior. Software should disable all debug events in order to prevent non-deterministic behavior on the trace interface to the core.

6. Clear DBSR register (initialize all debug event status)

   Although the PPC440EPx/GRx is defined to reset the DBSR debug event status bits during the reset operation (as specified in *Table 7-1* on page 172), this is not required by the architecture and hence the initialization software should not assume this behavior. Software should clear all such status in order to prevent non-deterministic behavior on the JTAG interface to the core.

7. Initialize CCR0 register
   1. Enable/disable broadcast of instructions to auxiliary processor (Disable to save power if not using FPU)
   2. Enable/disable broadcast of trace information (save power if not using trace instruction interface)
   3. Specify behavior for **icbt** and **dcbt**/**dcbtst** instructions
   4. Enable/disable gathering of separate store accesses
   5. Enable/disable hardware support for misaligned data accesses
   6. Enable/disable parity error recoverability (recoverability lowers load/store performance marginally.)
   7. Enable/disable cache read of parity bits depending on s/w compatibility requirements
8. Initialize CCR1 register
   1. Enable/disable full-line flushes as desired.
   2. Disable force cache-op miss (FCOM) and various parity error insertion (xxxPEI).
   3. Users may wish to initialize CCR1[TCS] here, or in the timer facilities section.
9. Configure instruction and data cache regions

   These steps must be performed prior to enabling the caches by setting the caching inhibited storage attribute of the corresponding TLB entry to 0.
   1. Clear the instruction and data cache normal victim index registers (INV0–INV3, DNV0–DNV3)
   2. Clear the instruction and data cache transient victim index registers (ITV0–ITV3, DTV0–DTV3)
   3. Set the instruction and data cache victim limit registers (IVLIM and DVLIM) according to the desired size of the normal, locked, and transient regions of each cache
10. Setup TLB entry to cover initial program memory page

    Since the PPC440EPx/GRx only initializes an architecturally-invisible shadow TLB entry during the reset operation, and since all shadow TLB entries are invalidated upon any context synchronization, special care must be taken during the initialization sequence to prevent any such context synchronizing operations (such as interrupts and the **isync** instruction) until after this step is completed, and an architected TLB entry has been established in the TLB. Particular care should be taken to avoid store operations, since write permission is disabled upon reset, and an attempt to execute any store operation would result in a Data Storage interrupt, thereby invalidating the shadow TLB entry.
    1. Initialize MMUCR
       - Specify TID field to be written to TLB entries

- Specify TS field to be used for TLB searches
- Specify store miss allocation behavior
- Enable/disable transient cache mechanism
- Enable/disable cache locking exceptions

2. Write TLB entry for initial program memory page
   - Specify EPN, RPN, ERPN, and SIZE as appropriate for system
   - Set valid bit
   - Specify TID = 0 (disable comparison to PID) or else initialize PID register to matching value
   - Specify TS = 0 (system address space) or else MSR[IS,DS] must be set to correspond to TS=1
   - Specify storage attributes (W, I, M, G, E, U0–U3) as appropriate for system
   - Enable supervisor mode fetch, read, and write access (SX, SR, SW)
3. Initialize PID register to match TID field of TLB entry (unless using TID = 0)
4. Setup for subsequent MSR[IS,DS] initialization to correspond to TS field of TLB entry
   (This is necessary only if the TS field of the TLB entry is being set to 1 and MSR[IS,DS] is already reset to 0)
   - Write new MSR value into SRR1
   - Write address from which to continue execution into SRR0
5. Setup for subsequent change in instruction fetch address

   Only necessary if EPN field of TLB entry changed from the initial value ($EPN_{0:19} \neq$ 0xFFFFF)
   - Write initial/new MSR value into SRR1
   - Write address from which to continue execution into SRR0
6. Fully initialize the TLB (tlbwe to all three words of each TLB entry; tlbre to TLB entries that are not fully initialized may result in parity exceptions).
7. Context synchronize to invalidate shadow TLB contents and cause new TLB contents to take effect
   - Use **isync** if not changing MSR contents and not changing the effective address of the rest of the initialization sequence
   - Use **rfi** if changing MSR to match new TS field of TLB entry (SRR1 will be copied into MSR, and program execution will resume at value in SRR0)
   - Use **rfi** if changing next instruction fetch address to correspond to new EPN field of TLB entry (SRR1 will be copied into MSR, and program execution will resume at value in SRR0)

   Instruction and data caches will now begin to be used, if the corresponding TLB entry has been setup with the caching inhibited storage attribute set to 0. Initialization software can now branch outside of the initial 4KB memory region as controlled by the address and size of the new TLB entry and/or any other TLB entries which have been setup.

11. Initialize interrupt resources
    1. Initialize IVPR to specify high-order address of the interrupt handling routines

       Make sure that the corresponding address region is covered by a TLB entry (or entries)
    2. Initialize IVOR0–IVOR15 registers (individual interrupt vector addresses)

       Make sure that the corresponding addresses are covered by a TLB entry (or entries)

Because the low order four bits of IVOR0–IVOR15 are reserved, the values written to those bits are ignored when the registers are written, and are read as zero when the registers are used. Therefore, all interrupt vector offsets are implicitly aligned on quadword boundaries. Software must take care to assure that all interrupt handlers are quadword-aligned.

3. Setup corresponding memory contents with the interrupt handling routines
4. Synchronize any program memory changes as required. (Refer to the *PPC440 Processor User's Manual* for more information on the instruction sequence necessary to synchronize changes to program memory prior to executing the new instructions.)

12. Configure debug facilities as desired
    1. Write DBCR1 and DBCR2 to specify IAC and DAC event conditions
    2. Clear DBSR to initialize IAC auto-toggle status
    3. Initialize IAC1–IAC4, DAC1–DAC2, DVC1–DVC2 registers to desired values
    4. Write MSR[DWE] to enable Debug Wait mode (if desired)
    5. Write DBCR0 to enable desired debug mode(s) and event(s)
    6. Context synchronize to establish new debug facility context (**isync**)
13. Configure timer facilities as desired
    1. Write DEC to 0 to prevent Decrementer exception after TSR is cleared
    2. Write TBL to 0 to prevent Fixed Interval Timer and Watchdog Timer exceptions after TSR is cleared, and to prevent increment into TBH prior to full initialization
    3. CCR1[TCS] (Timer Clock Select) can be initialized here, or earlier with the rest of the CCR1.
    4. Clear TSR to clear all timer exception status
    5. Write TCR to configure and enable timers as desired

       Software must take care with respect to the enabling of the Watchdog Timer reset function, as once this function is enabled, it cannot be disabled except by reset itself
    6. Initialize TBH value as desired
    7. Initialize TBL value as desired
    8. Initialize DECAR to desired value (if enabling the auto-reload function)
    9. Initialize DEC to desired value
14. Initialize facilities outside the processor core which are possible sources of asynchronous interrupt requests (including DCRs and/or other memory-mapped resources)

    This must be done prior to enabling asynchronous interrupts in the MSR
15. Initialize the MSR to enable interrupts as desired
    1. Set MSR[CE] to enable/disable Critical Input and Watchdog Timer interrupts
    2. Set MSR[EE] to enable/disable External Input, Decrementer, and Fixed Interval Timer interrupts
    3. Set MSR[DE] to enable/disable Debug interrupts
    4. Set MSR[ME] to enable/disable Machine Check interrupts

       Software should first check the status of the ESR[MCI] field and MCSR[MCS] field to determine whether any Machine Check exceptions have occurred after these fields are cleared by reset and before Machine Check interrupts are enabled (by this step). Any such exceptions set ESR[MCI] or MCSR[MCS] to 1, and this status can only be cleared explicitly by software. After the MCSR[MCS] field is known to be clear, the MCSR status bits (MCSR[1:8]) should be cleared by software to avoid possible confusion upon later ser-

vice of a machine check interrupt. Once MSR[ME] has been set to 1, subsequent Machine Check exceptions will result in a Machine Check interrupt.

5. Context synchronize to establish new MSR context (**isync**)

16. Initialize any other processor core resources as required by the system (GPRs, SPRGs, and so on)
17. Initialize any other facilities outside the processor core as required by the system
18. Initialize system memory as required by the system software

    Synchronize any program memory changes as required. (Refer to the *PPC440 Processor User's Manual* for more information on the instruction sequence necessary to synchronize changes to program memory prior to executing the new instructions)

19. Start the system software

    System software is generally responsible for initializing and/or managing the rest of the MSR fields, including:

    1. MSR[FP] to enable or disable the execution of floating-point instructions
    2. MSR[FE0,FE1] to enable/disable Floating-Point Enabled exception type Program interrupts
    3. MSR[PR] to specify user mode or supervisor mode
    4. MSR[IS,DS] to specify application address space or system address space for instructions and data
    5. MSR[WE] to place the processor into Wait State (halt execution pending an interrupt)

8. Bootstrap Controller

In preparation for booting, the PPC440EPx/GRx initializes register bit fields that affect system clocking, booting and other system features during a system reset. The initialization settings used in this process are determined by one of eight hardwired combinations on the three external bootstrap pins. See *Table 8-2*.

Two of the hardwired strap combinations cause the initialization setting to be read by the IIC bootstrap controller from a serial ROM attached to the IIC interface.

The other six hardwired bootstrap combinations cause the PPC440EPx/GRx to be initialized with predefined settings.

8.1 Bootstrap Configuration

Bootstrap values include clock ratios, PLL parameters, boot settings and several other configurations affecting PCI, NDFC, Ethernet and pin sharing. Options selected during reset are used to initialize registers throughout the PPC440EPx/GRx and are stored in the Serial Device Strap registers (SDR0_SDSTP0-SDR0_SDSTP3). *Table 8-1* lists the Serial Device Strap registers and the corresponding register bit fields initialized during reset.

Table 8-1. Bootstrap Register Bit Fields

Serial Device Strap Register	Serial Device Strap Register Bit Field	Bit Field Initialized During Reset	Description
SDR0_SDSTP0	SDR0_SDSTP0[ENG]	CPR0_PLLC0[ENG]	Engage
	SDR0_SDSTP0[SRC]	CPR0_PLLC0[SRC]	PLL Feedback Source
	SDR0_SDSTP0[SEL]	CPR0_PLLC0[SEL]	Feedback Selection
	SDR0_SDSTP0[TUNE]	CPR0_PLLC0[TUNE]	Tune bits
	SDR0_SDSTP0[FBDV]	CPR0_PLLD0[FBDV]	PLL Feedback Divisor
	SDR0_SDSTP0[FWDVA]	CPR0_PLLD0[FWDVA]	PLL Forward Divisor A
	SDR0_SDSTP0[FWDVB]	CPR0_PLLD0[FWDVB]	PLL Forward Divisor B
	SDR0_SDSTP0[PRBDV0]	CPR0_PRIMBD0[PRBDV0]	PLL Primary Divisor B
	SDR0_SDSTP0[OPBDV0]	CPR0_OPBD0[OPBDV0]	OPB Clock Divisor

Table 8-1. Bootstrap Register Bit Fields (continued)

Serial Device Strap Register	Serial Device Strap Register Bit Field	Bit Field Initialized During Reset	Description
SDR0_SDSTP1	SDR0_SDSTP1[LFBDV]	CPR0_PLLD0[LFBDV]	PLL Local Feedback Divisor
	SDR0_SDSTP1[PERDV0]	CPR0_PERD0[PERDV0]	Peripheral Clock Divisor 0
	SDR0_SDSTP1[MALD0]	CPR0_MALD[MALDV0]	MAL Clock Divisor 0
	SDR0_SDSTP1[SPCID0]	CPR0_SPCID[SPCID0]	PCI Clock Divisor
	SDR0_SDSTP1[PLLTIMER]		PLL TImer Length (CPR input)
	SDR0_SDSTP1[RW]	SDR0_EBC0[RW] EBC0_B0CR[BW]	EBC ROM Width
	SDR0_SDSTP1[RL]	SDR0_CP440[RL]	ROM location (Boot source)
	SDR0_SDSTP1[PAE]	SDR0_PCI0[PAE]	PCI Arbiter Enable
	SDR0_SDSTP1[PHCE]	SDR0_PCI0[PHCE] PCIC0_BRDGOPT2[HCE]	PCI Host Configuration Enable
	SDR0_SDSTP1[ZM]	SDR0_MFR0[ZM]	ZMII Mode
	SDR0_SDSTP1[CTE]	SDR0_PFC0[TRE]	CPU Trace Enable
	SDR0_SDSTP1[Nto1]	SDR0_CP440[Nto1]	CPU PLB N to 1 clock ratio
	SDR0_SDSTP1[PAME]	SDR0_PCI0[PAME]	PCI Asynchronous Mode Enable
	SDR0_SDSTP1[RSV]	Reserved	Reserved
SDR0_SDSTP2	SDR0_SDSTP2[MEN]	SDR0_CUST0[MEN]	Multiplex EMAC or NDFC
	SDR0_SDSTP2[NE]	SDR0_CUST0[NE] NDFC0_B0CR[EN]	NDFC Enable
	SDR0_SDSTP2[NBW]	SDR0_CUST0[NBW] NDFC0_B0CR[SZ]	NDFC Boot Width
	SDR0_SDSTP2[NBP]	SDR0_CUST0[NBP] NDFC0_CR[RPG]	NDFC Boot Page
	SDR0_SDSTP2[NBAC]	SDR0_CUST0[NBAC] NDFC0_CR[ARAC]	NDFC Boot Address Cycle
	SDR0_SDSTP2[NARE]	SDR0_CUST0[NARE] NDFC0_CR[ARE]	NDFC Auto read enable
	SDR0_SDSTP2[NRB]	SDR0_CUST0[NRB] NDFC0_CR[REN]	NDFC Ready/Busy bit
	SDR0_SDSTP2	SDR0_CUST0	NDFC Device Reset Count
	SDR0_SDSTP2[NCG]	SDR0_CUST0[NCG]	NDFC CS Gating
SDR0_SDSTP3	SDR0_SDSTP3[NDRSC]	SDR0_CUST1[NDRSC]	NDFC Device Read Count
	SDR0_SDSTP3[RSV]	SDR0_CUST1[RSV]	Reserved

8.2 Bootstrap Options

The PPC440EPx/GRx has eight hardwired bootstrap options described in *Table 8-2*. The bootstrap pins (UART0_DCD, UART0_DSR and UART0_CTS) select the boot option at reset. The six options A, B, C, D, E, and F provide predefined bootstrap configurations. Options G and H enable the IIC bootstrap controller and select the IIC address, 0b10101000 (0xA8) or 0b10100100 (0xA4) respectively.

The six hardwired bootstrap options A, B, C, D, E, and F include configurations for booting from either an EBC, PCI, or NDFC attached ROM. Options A, B, and C support booting from an EBC attached ROM with 8- or 16-bit data widths. Options E and F support booting over PCI. Option D supports booting over NDFC. *Table 8-3* contains descriptions of each bootstrap option.

If options G or H are wired, the controller reads 16 bytes from a serial ROM accessible from the IIC0 interface and stores them in the Serial Device Strap Registers. If the IIC bootstrap controller is unable to successfully read data from the serial ROM, the PPC440EPx/GRx defaults to Bootstrap Option A.

If the IIC bootstrap controller is disabled (strapping pins are wired for one of the options A—F), the PPC440EPx/GRx is not limited to the selected bootstrap configuration. After system reset, software can change the initilization settings by following the procedure described below.

1. Check the following registers for the desired configuration:
   - CPR0_PLLC0
   - CPR0_PLLD0
   - CPR0_PRIMAD0
   - CPR0_PRIMBD0
   - CPR0_OPBD0
   - CPR0_PERD0
   - CPR0_MALD
   - CPR0_SPCID

   If these registers are correctly configured, no further action is required and steps 2 and 3 need not be done. Otherwise, set the desired configuration and proceed with steps 2 and 3.
2. Set CPR0_ICFG[RLI] = 1. The Reload Inhibit bit preserves the configuration set in step 1 through a chip reset. The preserved register contents are used as the boot configuration after the chip reset.
3. Set DBCR0[RST] = 0b10 to generate a chip reset.

Note: CPR0_ICFG[RLI] does not preserve the content of SDR0_CP440[Nto1] through a chip reset. If changes to the clocks alter the CPU:PLB clock ratio from N:1 to N:2, this procedure cannot be used since a change of SDR0_CP440[Nto1] is required.

Table 8-2. Bootstrap Pins

UART0_DCD	UART0_DSR	UART0_CTS	Description
0	0	0	Bootstrap Option A SysClk - 33MHz VCO - 666MHz CPU - 333MHz PLB - 133MHz OPB - 66MHz PCI - 33MHz PerClk - 66MHz Boot ROM Location - EBC Boot width - 8 bits

Table 8-2. Bootstrap Pins (continued)

UART0_DCD	UART0_DSR	UART0_CTS	Description
0	0	1	Bootstrap Option B SysClk - 33MHz VCO - 666MHz CPU - 333MHz PLB - 133MHz OPB - 66MHz PCI - 33MHz PerClk - 66MHz Boot ROM Location - EBC Boot width - 16 bits
0	1	0	Bootstrap Option C SysClk - 33MHz VCO - 1066MHz CPU - 533MHz PLB - 133MHz OPB - 66MHz PCI - 33MHz PerClk - 66MHz Boot ROM Location - EBC Boot width - 16 bits
0	1	1	Bootstrap Option D SysClk - 33MHz VCO - 666MHz CPU - 333MHz PLB - 133MHz OPB - 66MHz PCI - 33MHz PerClk - 66MHz Boot ROM Location - NAND Flash Boot width - 8 bits (NAND Flash addressing mode: four cycles, 512B per page)
1	0	0	Bootstrap Option E SysClk - 33MHz VCO - 666MHz CPU - 333MHz PLB - 133MHz OPB - 66MHz PCI - 33MHz PerClk - 66MHz Boot ROM Location - PCI Boot width - na
1	0	1	Bootstrap Option G IIC Bootstrap controller enabled, serial ROM address 0b10101000 (0xA8)
1	1	0	Bootstrap Option F SysClk - 33MHz VCO - 1066MHz CPU - 533MHz PLB - 133MHz OPB - 66MHz PCI - 66MHz PerClk - 66MHz Boot ROM Location - PCI Boot width - na

Table 8-2. Bootstrap Pins (continued)

UART0_DCD	UART0_DSR	UART0_CTS	Description
1	1	1	Bootstrap Option H IIC Bootstrap controller enabled, serial ROM address 0b10100100 (0xA4)

Table 8-3. Bootstrap Configurations

Serial Device Strap Register Bit Field	Bootstrap Option						Description
	A	B	C	D	E	F	
SDR0_SDSTP0[ENG]	1	1	1	1	1	1	Engage 1 PLL's VCO is the source for the PLL forward dividers
SDR0_SDSTP0[SRC]	0	0	0	0	0	0	PLL feedback source 0 Feedback originates from PLLOutA 1 Feedback originates from PLLOutB
SDR0_SDSTP0[SEL]	000	000	000	000	000	000	Feedback selection 000 PLL output (A or B)
SDR0_SDSTP0[TUNE]	10001 11000	10001 11000	11101 11100	10001 11000	10001 11000	11101 11100	PLL tune bits 0100110110 $9 < M \leq 10$ 1100111100 $22 < M \leq 40$
SDR0_SDSTP0[FBDV]	01010	01010	01000	01010	01010	01000	PLL Feedback Divisor 01100 PLL Feedback Divisor = 12 00101 PLL Feedback Divisor = 5 00011 PLL Feedback Divisor = 3
SDR0_SDSTP0[FWDVA]	0010	0010	0010	0010	0010	0010	PLL Forward Divisor A 0010 PLL Forward Divisor A = 2
SDR0_SDSTP0[FWDVB]	001	001	010	001	001	010	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 011 PLL Forward Divisor B = 3 101 PLL Forward Divisor B = 5
SDR0_SDSTP0[PRBDV0]	101	101	100	101	101	100	PLL Primary Divisor B 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 **Note:** Reset value for PLL Primary Divisor A is 1.
SDR0_SDSTP0[OPBDV0]	10	10	10	10	10	10	OPB clock divisor 0 10 OPB Clock Divisor 0 = 2
SDR0_SDSTP1[LFBDV]	000001	000001	000010	000001	000001	000010	PLL Local Feedback Divisor 000001 PLL local feedback divisor = 1
SDR0_SDSTP1[PERDV0]	010	010	010	010	010	010	Peripheral clock divisor 0 010 Peripheral clock divisor = 2 100 Peripheral clock divisor = 4
SDR0_SDSTP1[MALDV0]	10	10	10	10	10	10	MAL clock divisor 0 10 MAL clock divisor 0 = 2
SDR0_SDSTP1[SPCID0]	00	00	00	00	00	10	SPCID0 10 (divide by 2)

Table 8-3. Bootstrap Configurations (continued)

Serial Device Strap Register Bit Field	Bootstrap Option						Description
	A	B	C	D	E	F	
SDR0_SDSTP1[PLLTIMER]	1111	1111	1111	1111	1111	1111	Timer setting for PLL locking. Set the four upper bit of timer length value. The lower bits are 1
SDR0_SDSTP1[RW]	00	01	01	10	10	01	ROM boot width 00 ROM Boot Width = 8 bit 01 ROM Boot Width = 16 bit 10 ROM Boot Width = 32bit
SDR0_SDSTP1[RL]	00	00	00	10	01	01	ROM location 00 EBC 01 PCI 10 NDFC
SDR0_SDSTP1[PAE]	0	0	0	0	0	0	PCI internal arbiter enable 0 Disable
SDR0_SDSTP1[PHCE]	0	0	0	0	0	0	PCI host configuration enable 0 Disable
SDR0_SDSTP1[ZM]	00	00	00	00	00	00	ZMII mode 00 MII mode
SDR0_SDSTP1[CTE]	0	0	0	0	0	0	CPU Trace enable 0 Disable
SDR0_SDSTP1[N to 1]	0	0	1	0	0	1	CPU/PLB ration N/P 0 not N to 1 1 N to 1
SDR0_SDSTP1[PAME]	1	1	1	1	1	1	PCI asynchronous mode enable 1 Enable
SDR0_SDSTP1[RSV]							Reserved (4 bits)
SDR0_SDSTP2[MEN]	01	01	01	10	01	01	Multiplex NDFC, or GPIO 01 high Z 10 NDFC 11 GPIO
SDR0_SDSTP2[NE]	0	0	0	1	0	0	NDFC enable 0 Disable 1 Enable
SDR0_SDSTP2[NBW]	0	0	0	0	0	0	NDFC boot width 0 8 bits
SDR0_SDSTP2[NBP]	0000	0000	0000	0000	0000	0000	NDFC boot page selection
SDR0_SDSTP2[NBAC]	00	00	00	01	00	00	NDFC boot address selection cycle 00 Three cycles, 512B per page 01 Four cycles, 512B per page
SDR0_SDSTP2[NARE]	0	0	0	1	0	0	NDFC Auto-Read Enable 0 Disable 1 Enable
SDR0_SDSTP2[NRB]	0	0	0	0	0	0	NDFC Ready/Busy 0 Ready/Busy disable 1 Ready/Busy enable

Table 8-3. Bootstrap Configurations (continued)

Serial Device Strap Register Bit Field	Bootstrap Option						Description
	A	B	C	D	E	F	
SDR0_SDSTP2[NDRSC]	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	1000 0010 0011 0101	NDFC Device Reset Counter
SDR0_SDSTP2[NCG]	0000	0000	0000	1000	0000	0000	NDFC/EBC Chip select Gating
SDR0_SDSTP3[NDRDC]	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	0000 1101 0000 0101	NDFC Device Read Count
SDR0_SDSTP3[RSV]							Reserved (16 bits)
Note: If CPR0_ICGF[RLI] = 0, the reset value for PLL Primary Divisor A is 1 (CPR0_PRIMAD[PRADV0] = 1).							

8.3 IIC Bootstrap Operations

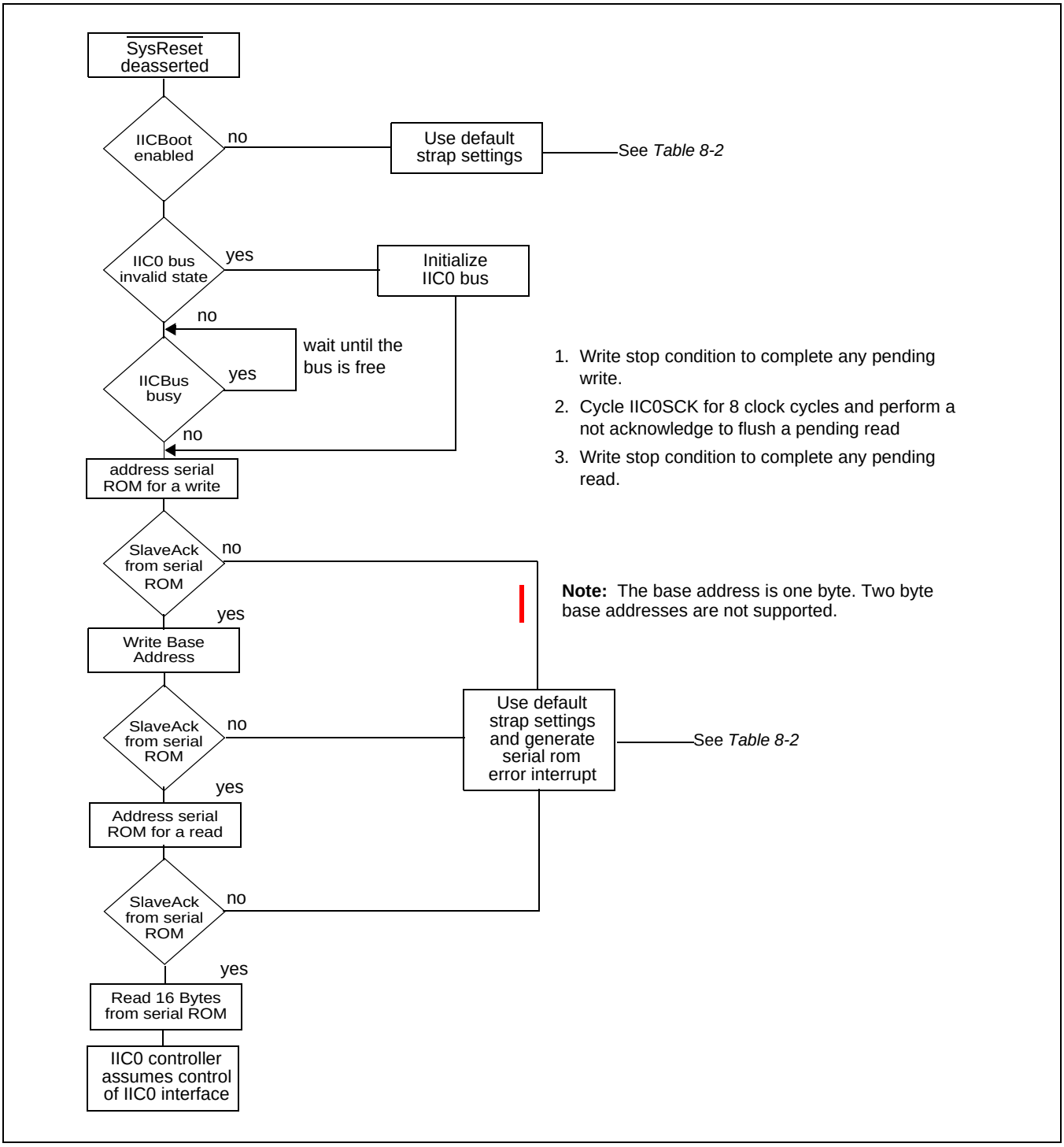
After the deassertion of $\overline{\text{SysReset}}$, the IIC bootstrap controller tests the IIC bus to determine if the bus is in an invalid state or busy. If the bus is in an invalid state, it places the IIC0 bus in a known state through an initialization sequence and accesses the serial ROM as described in *Figure 8-1*. If the bus is busy, it waits until the bus is free before accessing the serial ROM. To access the ROM, the IIC bootstrap controller writes the base address (0x00) and then reads 16 bytes starting from the base address 0x00. The data in the serial ROM is organized as shown in *Table 8-4*.

If the IIC bootstrap controller is faster than the serial ROM, then the later can hold the IIC0 clock signal low until it is prepared for the next transaction. If the serial ROM does not acknowledge its address or the receipt of the base address, the PPC440EPx/GRx defaults to Bootstrap Option A in *Table 8-3* and generates a serial ROM error interrupt, UIC1_SR [SRE]. Once interrupts are enabled, initialization software should check UIC1_SR[SRE] to ensure no errors occurred while accessing the serial ROM.

Note: The IIC bootstrap controller assumes the serial ROM has one byte base address.

Figure 8-1 shows the IIC bootstrap controller flow.

Figure 8-1. IIC Bootstrap Controller Flow


SysReset deasserted
IICBoot enabled
no
Use default strap settings
See Table 8-2
IIC0 bus invalid state
yes
Initialize IIC0 bus
no
wait until the bus is free
IICBus busy
yes
no
address serial ROM for a write
SlaveAck from serial ROM
no
yes
Write Base Address
SlaveAck from serial ROM
no
Use default strap settings and generate serial rom error interrupt
See Table 8-2
yes
Address serial ROM for a read
SlaveAck from serial ROM
no
yes
Read 16 Bytes from serial ROM
IIC0 controller assumes control of IIC0 interface
1. Write stop condition to complete any pending write.
2. Cycle IIC0SCK for 8 clock cycles and perform a not acknowledge to flush a pending read
3. Write stop condition to complete any pending read.
Note: The base address is one byte. Two byte base addresses are not supported.

8.3.1 Performance

The IIC bootstrap controller requires 188 IIC0_SCK clock cycles to read 16 bytes, 53 clock cycles for the first byte and 9 cycles for each of the remaining 15 bytes. The IIC0_SCK clock is generated directly from SysClk during system reset by dividing it by 768 (IIC0_SCK freq = SysClk/768).

Assuming a SysClk of 66.66 MHz, the time needed to read the serial ROM is 2.1 mS. Serial ROM read time = 188*768/(SysClk freq)

Table 8-4. Serial ROM Memory Map

ROM Byte Address	Bit Number Within a Byte	Register[Field]
0	7 (MSb)	SDR0_SDSTP0[ENG]
	6	SDR0_SDSTP0[SRC]
	5:3	SDR0_SDSTP0[SEL]
	2:0 (LSb)	SDR0_SDSTP0[$TUNE_{5:7}$]
1	7:1	SDR0_SDSTP0[$TUNE_{8:14}$]
	0	SDR0_SDSTP0[$FBDV_{15}$]
2	7:4	SDR0_SDSTP0[$FBDV_{16:19}$]
	3:0	SDR0_SDSTP0[FBDVA]
3	7:5	SDR0_SDSTP0[FBDVB]
	4:2	SDR0_SDSTP0[PRBDV0]
	1:0	SDR0_SDSTP0[OPBDV0]
4	7:2	SDR0_SDSTP1[LFBDV]
	1:0	SDR0_SDSTP1[$PERDV0]_{0:1}$
5	7	SDR0_SDSTP1[$PERDV0]_2$
	6:5	SDR0_SDSTP1[MALDV0]
	4:3	SDR0_SDSTP1[PCID0]
	2:0	SDR0_SDSTP1[$PLLTIMER]_{0:2}$
6	7	SDR0_SDSTP1[$PLLTIMER]_3$
	6:5	SDR0_SDSTP1[RW]
	4:3	SDR0_SDSTP1[RL]
	2	SDR0_SDSTP1[PAE]
	1	SDR0_SDSTP1[PHCE]
	0	SDR0_SDSTP1[$ZM]_0$
7	7	SDR0_SDSTP1[$ZM]_1$
	6	SDR0_SDSTP1[CTE]
	5	SDR0_SDSTP1[Nto1]
	4	SDR0_SDSTP1[PAME]
	3:0	SDR0_SDSTP1[RSV]
8	7:6	SDR0_SDSTP2[MEN]
	5	SDR0_SDSTP2[NE]
	4	SDR0_SDSTP2[NBW]
	3:0	SDR0_SDSTP2[NBP]

Table 8-4. Serial ROM Memory Map (continued)

ROM Byte Address	Bit Number Within a Byte	Register[Field]
9	7:6	SDR0_SDSTP2[NBAC]
	5	SDR0_SDSTP2[NARE]
	4	SDR0_SDSTP2[NRB]
	3:0	SDR0_SDSTP2[NDRSC]$_{0:3}$
A	7:0	SDR0_SDSTP2[NDRSC]$_{4:11}$
B	7:4	SDR0_SDSTP2[NDRSC]$_{12:15}$
	3:0	SDR0_SDSTP2[NCG]
C	7:0	SDR0_SDSTP3[NDRDC]$_{0:7}$
D	7:0	SDR0_SDSTP3[NDRDC]$_{8:15}$
E	7:0	SDR0_SDSTP3[RSV]$_{0:7}$
F	7:0	SDR0_SDSTP3[RSV]$_{8:15}$

8.4 IIC Bootstrap Registers

All bootstrap registers are accessed indirectly through the SDR0_CFGADDR and SDR0_CFGDATA registers using the mtdcr and mfdcr instructions. *Table 8-5* lists the DCR addresses for the SDR0_CFGADDR and SDR0_CFGDATA registers.

Table 8-5. System DCR Register

Register	DCR Address	Access	Description
SDR0_CFGADDR	0x00E	R/W	System DCR Address Register
SDR0_CFGDATA	0x00F	R/W	System DCR Data Register

To read or write one of the bootstrap registers, software first writes the register address offset of the target register into the SDR0_CFGADDR register. The target register can then be read or written through the SDR0_CFGDATA DCR register. The following PowerPC code illustrates this procedure by reading the SDR0_SDSTP0 register.

```
li      r3, SDR0_CFGADDR      ! address offset of SDR0_CFGADDR
mtdcr   SDR0_CFGDATA, r3      ! set offset address
mfdcr   r4, SDR0_CFGDATA      ! read value of SDR0_CFGDATA
```

Note: *Table 8-6* lists the device control registers that are directly affected by the bootstrap initialization process. Other system device control registers that control PPC440EPx/GRx miscellaneous devices are described in *Reset and Initialization* on page 169 while the clocking and power-on reset registers are described in *Clocking Registers* on page 284.

Table 8-6. IIC Bootstrap Controller Registers

Mnemonic	Register	Address	Access	Page
SDR0_PINSTP	Pin Strapping Register	0x0040	R	197
SDR0_SDCS0	Serial Device Controller Settings Register	0x0060	R	197
SDR0_SDSTP0	Serial Device Strap Register 0	0x0020	R	198
SDR0_SDSTP1	Serial Device Strap Register 1	0x0021	R	200
SDR0_SDSTP2	Read-Only Version of SDR0_CUST0	0x4001	R	201
SDR0_SDSTP3	Read-Only Version of SDR0_CUST1	0x4003	R	202
SDR0_CUST0	Customer Configuration Register 0	0x4000	R/W	203
SDR0_CUST1	Customer Configuration Register 1	0x4002	R/W	204
SDR0_EBC0	EBC Configuration Register	0x0100	R/W	204

8.4.1 Pin Strapping Register (SDR0_PINSTP)

The Pin Strapping Register (SDR0_PINSTP) records how the bootstrap pins are strapped during system reset. A copy of pin strap status is also maintained by *Initial Configuration Register (CPR0_ICFG)* on page 176.

Figure 8-2. Pin Strapping Register (SDR0_PINSTP)

0	U0DCD	$\overline{\text{UART0\_DCD}}$ Strapping 0 Strapped low 1 Strapped high	
1	U0DSR	$\overline{\text{UART0\_DSR}}$ Strapping 0 Strapped low 1 Strapped high	
2	U0CTS	$\overline{\text{UART0\_CTS}}$ Strapping 0 Strapped low 1 Strapped high	
3:31		Reserved	

8.4.2 Serial Device Controller Settings Register (SDR0_SDCS0)

The Serial Device Controller Settings Register contains the serial ROM address and status showing if the IIC Bootstrap Controller was enabled during system reset.

Figure 8-3. Serial Device Controller Settings Register (SDR0_SDCS0)			
0:6	SBA	SEPROM Base Address: UART0_DSR=0 Base address 0xA8 0b1010100 UART0_DSR=1 Base address 0xA4 0b1010010	
7:29		Reserved	
30	SDC	Serial Device Control 0 Serial device control disabled 1 Serial device control enabled	
31	SDD	Serial Device Detection 0 Serial device detection disabled 1 Serial device detection enabled	SDR0_SDCR[SDD] = 1 if SDR0_SDCR[SDC] = 1

8.4.3 Serial Device Strap Register 0 (SDR0_SDSTP0)

SDR0_SDSTP0 is 32-bit read-only register. This register is reserved for system PLL and configuration information. The SDR0_SDSTP0 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap options described in *Bootstrap Options* on page 189. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, the bootstrap option A is used as the default configuration.

Figure 8-4. Serial Device Strap Register 0 (SDR0_SDSTP0)			
0	ENG	Engage 0 SysClk is the source for PLL forward dividers. 1 PLL's VCO is the source for PLL forward dividers.	
1	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB	
2:4		Reserved	Only the default value 0b000 is allowed. Other values are for manufacturing use only.
5:14	TUNE	TUNE bits	
15:19	FBDV	PLL Feedback Divisor 00000 PLL Feedback Divisor = 32 00001 PLL Feedback Divisor = 1 00010 PLL Feedback Divisor = 2 00011 PLL Feedback Divisor = 3 00100 PLL Feedback Divisor = 4 .... 11111 PLL Feedback Divisor = 31	

Figure 8-4. Serial Device Strap Register 0 (SDR0_SDSTP0) (continued)			
20:23	FWDVA	PLL Forward Divisor A 0000 PLL Forward Divisor A = 16 0001 PLL Forward Divisor A = 1 0010 PLL Forward Divisor A = 2 0011 PLL Forward Divisor A = 3 0100 PLL Forward Divisor A = 4 0101 PLL Forward Divisor A = 5 0110 PLL Forward Divisor A = 6 0111 PLL Forward Divisor A = 7 1000 PLL Forward Divisor A = 8 1001 Reserved 1010 PLL Forward Divisor A = 10 1011 Reserved 1100 PLL Forward Divisor A = 12 1101 Reserved 1110 PLL Forward Divisor A = 14 1111 Reserved	
24:26	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 Reserved 010 PLL Forward Divisor B = 2 011 PLL Forward Divisor B = 3 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 5 110 PLL Forward Divisor B = 6 111 PLL Forward Divisor B = 7	
27:29	PRBDV0	PLL Primary Divisor B 000 PLL Primary Divisor B = 8 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 011 PLL Primary Divisor B = 3 100 PLL Primary Divisor B = 4 101 PLL Primary Divisor B = 5 110 PLL Primary Divisor B = 6 111 PLL Primary Divisor B = 7	**Note:** If CPR0_ICFG[RLI] = 0, then the reset value for PLL Primary Divisor A is 1 (CPR0_PRIMAD[PRADV0]=1). PRBDV0 = 1 is the recommended setting. This setting ensures, PLLOUTB clock does not exceed the rated speed of the processor core (CPU Clk). The output of FWDVA which is PLLOUTB is the same frequency as the PLB CLK when PRBDV0 is set to 1.
30:31	OPBDV0	OPB Clock Divisor 0 00 OPB clock divisor 0 = 4 01 OPB clock divisor 0 = 1 10 OPB clock divisor 0 = 2 11 OPB clock divisor 0 = 3	

8.4.4 Serial Device Strap Register 1 (SDR0_SDSTP1)

SDR0_SDSTP1 is 32-bit read-only register. This register is reserved for system PLL and configuration information. The SDR0_SDSTP1 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap options described in *Bootstrap Options* on page 189. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, the bootstrap option A is used as the default configuration.

Figure 8-5. Serial Device Strap Register 1 (SDR0_SDSTP1)

0:5	LFBDV	PLL Local Feedback Divisor 00_0000 PLL local feedback divisor = 64 00_0001 PLL local feedback divisor = 1 00_0010 PLL local feedback divisor = 2 .... 11_1111 PLL local feedback divisor = 63	The LFBDV is outside of PLL and is used to allow the PLL to lock using feedback directly from a PLL output. Program this divider to a value equivalent to the divide from the PLL to some clock to allow the PLL to be switched to use feedback from that clock. This allows the PLL to compensate for the latency associated with that clock's tree.
6:8	PERDV0	Peripheral Clock Divisor 0 000 Reserved 001 Peripheral clock divisor 0 = 1 010 Peripheral clock divisor 0 = 2 011 Peripheral clock divisor 0 = 3 100 Peripheral clock divisor 0 = 4 101 Peripheral clock divisor 0 = 5 110 Reserved 111 Reserved	
9:10	MALDV0	MAL Clock Divisor 0 00 MAL clock divisor 0 = 4 01 MAL clock divisor 0 = 1 10 MAL clock divisor 0 = 2 11 MAL clock divisor 0 = 3	Set MALDV0 = OPBDV0.
11:12	SPCID0	PCI Clock Divisor 0 00 PCI clock divisor 0 = 4 01 PCI clock divisor 0 = 1 10 PCI clock divisor 0 = 2 11 PCI clock divisor 0 = 3	
13:16	TS	Timer Setting (4 upper bits of the counter)	Length of timer for PLL locking. See also IIC bytes 5 & 6.
17:18	RW	EBC ROM Width 00 8-bit 01 16-bit 10 32-bit (Mandatory for Boot from Nand Flash) 11 Reserved	
19:20	RL	ROM Location 00 EBC 01 PCI 10 NDFC 11 Reserved	
21	PAE	PCI Internal Arbiter Enable 0 Disable 1 Enable	
22	PHCE	PCI Host Configuration Enable 0 Disable 1 Enable	

Figure 8-5. Serial Device Strap Register 1 (SDR0_SDSTP1) (continued)			
23:24	ZMII	ZMII mode 00 MII mode 01 SMII mode 10 Reserved 11 Reserved	
25	CTE	CPU trace enable 0 Disable 1 Enable	
26	Nto1	CPU/PLB ration N/P 0 not N to 1 1 N to 1	
27	PAME	PCI Asynchronous mode enable 0 Reserved 1 Enable	PAME = 0 is not supported.
28:31		Reserved	

8.4.5 Read-Only Equivalent of SDR0_CUST0 (SDR0_SDSTP2)

SDR0_SDSTP2 is a 32-bit read-only register. This register is reserved for user defined initialization data. The SDR0_SDSTP2 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap options described in *Bootstrap Options* on page 189. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, the bootstrap option A is used as the default configuration. Changes to SDR0_SDSTP2 are copied into SDR0_CUST0 following a reset.

Figure 8-6. Serial Device Strap Register 2 (SDR0_SDSTP2)			
0:1	MEN	Multiplex NDFC or GPIO 00 Reserved 01 Reserved 10 NDFC 11 GPIO	
2	NE	NDFC Enable 0 Disabled 1 Enabled	
3	NBW	NDFC Boot Width 0 8-bit 1 16-bit	Width of external device.
4:7	NBP	NDFC Boot Page selection	Typically, page 0 is selected.
8:9	NBAC	NDFC Boot Address Selection Cycle 00 3 cycles, 1 Col. + 2 Row (512B page) 01 4 cycles, 1 Col. + 3 Row (512B page) 10 4 cycles, 2 Col. + 2 Row (2KB page) 11 5 cycles, 2 Col. + 3 Row (2KB page)	
10	NARE	NDFC Auto-Read Enable 0 Disabled 1 Enabled	When booting from NAND flash, set NARE = 1.

Figure 8-6. Serial Device Strap Register 2 (SDR0_SDSTP2) (continued)			
11	NRB	NDFC Ready/Busy 0 Ready/Busy disable 1 Ready/Busy enable	When booting from NAND flash, set NRB = 0. When NRB = 1, the NDFC waits for NFRdyBusy to be sampled high before allowing read cycles to occur.
12:27	NDRSC	NDFC Device Reset Counter value Number PerClk cycles NDFC waits for I/O signal NFRdyBusy = 1 after reset.	Many NAND flash devices specify a worst-case reset time of 500µs. **Example:** When PerClk = 33.33MHz (30ns) then NDRSC = 0x411B = 16667 cycles = 500µs
28	NCG0	NDFC /EBC Chip select (enable) gating Bit 28: CS0 (0 = EBC, 1= NDFC)	Select either the EBC or the NDFC to control chip select (enable) pins. **Note:** $\overline{\text{PerCSx}}$ (chip select) for EBC or $\overline{\text{NFCEx}}$ (chip enable) for NDFC
29	NCG1	NDFC /EBC Chip select (enable) gating Bit 29: CS1 (0 = EBC, 1= NDFC)	
30	NCG2	NDFC /EBC Chip select (enable) gating Bit 30: CS2(0 = EBC, 1= NDFC)	
31	NCG3	NDFC /EBC Chip select (enable) gating Bit 31: CS3 (0 = EBC, 1= NDFC)	

8.4.6 Read-Only Version of SDR0_CUST1 (SDR0_SDSTP3)

SDR0_SDSTP3 is a 32-bit read-only register. This register is reserved for user defined initialization data. The SDR0_SDSTP3 is reset according to values read by the IIC Bootstrap controller or provided by bootstrap options described in *Bootstrap Options* on page 189. If the IIC Bootstrap controller is unable to read the boot-strap information from a serial ROM device, the bootstrap option A is used as the default configuration. Changes to SDR0_SDSTP3 are copied into SDR0_CUST1 following a reset.

Figure 8-7. Serial Device Strap Register 3 (SDR0_SDSTP3)			
0:15	NDRDC	Device Read Count Maximum number of PerClk cycles NDFC waits for NFRdyBusy = 1.	Set NDRSC to allow 50µs for a page read command to complete. When NFRdyBusy goes high before reaching the NDRSC count, the counter resets. The EBC transfer wait state timer (EBC0_B0AP[TWT]) starts counting wait states when either NFRdyBusy goes high or the count exceeds NDRSC. **Example:** For PerClk = 33.33MHz (30ns), set NDRSC = 0x0683 = 1667 cycles = 50µs
16:31		Reserved	

8.4.7 Customer Configuration Register 0 (SDR0_CUST0)

The Custom Configuration Register 0 (SDR0_CUST0) contains user defined initialization data. The bootstrap controller reads 8 bytes of user defined bootstraps and stores them in the SDR0_SDSTP2 register. The contents of read-only register SDR0_SDSTP2 initializes SDR0_CUST0.

Figure 8-8. Serial Device Strap Register 0 (SDR0_CUST0)

0:1	MEN	Multiplex NAND Flash or GPIO 00 Reserved 01 Reserved 10 NDFC 11 GPIO	[GPIO22]$\overline{\text{NFRdyBusy}}$ [GPIO12]$\overline{\text{NFREn}}$ [GPIO14]NFCLE [GPIO15]$\overline{\text{NFALE}}$ [GPIO13]$\overline{\text{NFWEn}}$
2	NE	NDFC Enable 0 Disabled 1 Enabled	
3	NBW	NDFC Boot Width 0 8-bit 1 16-bit	Width of external device.
4:7	NBP	NDFC Boot Page Selection	
8:9	NBAC	NDFC Boot Address Selection Cycle	
10	NARE	NDFC Auto Read Disable 0 Disabled 1 Enabled	
11	NRB	NDFC Ready/Busy 0 Ready/Busy disable 1 Ready/Busy enable	
12:27	NDRSC	NDFC Device Reset Counter Value	
28	NCG0	NDFC /EBC Chip select (enable) gating Bit 28: CS0 (0 = EBC, 1= NDFC)	Select either the EBC or the NDFC to control chip select (enable) pins. **Note:** $\overline{\text{PerCSx}}$ (chip select) for EBC or $\overline{\text{NFCEx}}$ (chip enable) for NDFC
29	NCG1	NDFC /EBC Chip select (enable) gating Bit 29: CS1 (0 = EBC, 1= NDFC)	
30	NCG2	NDFC /EBC Chip select (enable) gating Bit 30: CS2(0 = EBC, 1= NDFC)	
31	NCG3	NDFC /EBC Chip select (enable) gating Bit 31: CS3 (0 = EBC, 1= NDFC)	

8.4.8 Custom Configuration Register 1 (SDR0_CUST1)

The Custom Configuration Register (SDR0_CUST1) contains user defined initialization data. The bootstrap controller reads bytes of user defined bootstraps and stores them in the SDR0_SDSTP3 register. The contents of read-only register SDR0_SDSTP3 initializes SDR0_CUST1.

Figure 8-9. Custom Configuration Register 1 (SDR0_CUST1)

0:15	NDRDC	Device Read Count	
16:31		Reserved	Reserved for user's data

8.4.9 EBC Configuration Register (SDR0_EBC0)

The EBC Configuration Register (SDR0_EBC0) is initialized with values provided by the SDR0_SDSTP1[RW] bit field. This register configures EBC bank 0 peripheral width when booting from an EBC attached ROM.

Figure 8-10. EBC Configuration Register (SDR0_EBC0)

0:1		Reserved	Reset value = 0
2:3	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 32-bit ROM 11 Reserved	When booting from NDFC, this value should be 10, as in boot configuration C. Since the actual device is hidden from the EBC, 32 bits indicates the width of the controller itself. Reset value = SDR0_SDSTP1[RW]
3:31		Reserved	Reset value = 0

9. Universal Interrupt Controller

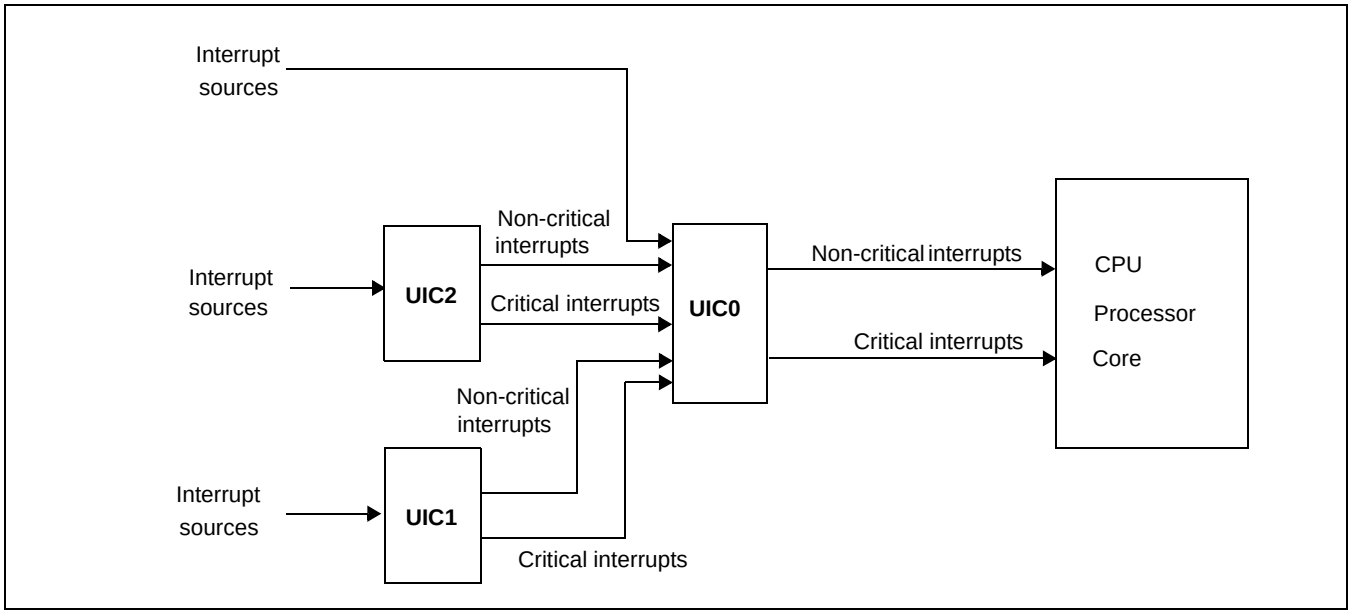
The PPC440EPx/GRx contains three universal interrupt controllers (UIC0, UIC1, and UIC2) that provide all necessary control, status, and communication between the various internal and external interrupt sources and the processor core.

9.1 UIC Overview

The UICs support 68 internal interrupts and 10 external interrupts. Status reporting (using the UIC Status Register [UIC*x*_SR]) is provided to ensure that systems software can determine the current and interrupting state of the system and respond appropriately. Software can generate interrupts to simplify software development and for diagnostics.

UIC0 collects interrupts from internal and external sources, including the critical and non-critical interrupt outputs of the secondary interrupt controllers, UIC1 and UIC2. The UICs are cascaded as shown in *Figure 9-1* below:

Figure 9-1. Cascaded UIC Organization



The interrupts can be programmed, using the UIC Critical Register (UIC*x*_CR), to generate either a critical or a non-critical interrupt signal to the processor core.

The privileged **mtdcr** and **mfdcr** instructions, which are used by system software, are used to read and write the UIC registers.

An optional critical interrupt vector generator can reduce interrupt handling latency for critical interrupts. Vector calculation is described in detail in *UIC0 Vector Register (UIC0_VR)* on page 244.

9.2 UIC Features

- Support for 68 internal and 10 external interrupts
- Support for asynchronous level- or edge-sensitive interrupt types
- Programmable polarity for all interrupt types

- Programmable critical/non-critical interrupt selection for each interrupt bit
- Prioritized critical interrupt vector generation
- A UIC Status Register (UIC*x*_SR) providing the following information:
  - Current state of interrupts
  - Current state of all enabled interrupts (those masked using the UIC Enable Register (UIC*x*_ER))

9.3 UIC Interrupt Assignments

The UIC supports internal and external interrupt sources as shown in *Table 9-1*, *Table 9-2*, and *Table 9-3*.

Table 9-1. UIC0 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Level	UART0
1	High	Level	UART1
2	High	Level	IIC0
3	High	Level	Kasumi Ready for Data
4	High	Level	Kasumi Data Available
5	High	Level	PCI Command Write
6	High	Level	PCI Power Management
7	High	Level	IIC1
8	High	Level	SPI
9	Low	Level	External PCI SERR
10	High	Level	MAL TX EOB
11	High	Level	MAL RX EOB
12	High	Level	DMA Channel 0 (DMA2P30)
13	High	Level	DMA Channel 1 (DMA2P30)
14	High	Level	DMA Channel 2 (DMA2P30)
15	High	Level	DMA Channel 3 (DMA2P30)
16	High	Level	UDMA IRQ 0 (DMA2P40)
17	High	Level	UDMA IRQ 1 (DMA2P40)
18	High	Level	UDMA IRQ 2 (DMA2P40)
19	High	Level	UDMA IRQ 3 (DMA2P40)
20	Low	Edge	USB2.0 Device *(PPC440EPX only)*
21	Low	Level	USB2.0 Host OHCI IRQ 1 *(PPC440EPX only)*
22	Low	Level	USB2.0 Host OHCI IRQ 2 (Not used) *(PPC440EPX only)*
23	High	Level	Security Function
24	High	Level	EMAC0
25	High	Level	EMAC1
26	High	Level	USB2.0 Host EHCI *(PPC440EPX only)*
27	Programmable	Programmable	External IRQ 4
28	High	Level	UIC2 Non-critical interrupt
29	High	Level	UIC2 Critical interrupt
30	High	Level	UIC1 Non-Critical interrupt
31	High	Level	UIC1 Critical Interrupt

Table 9-2. UIC1 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Level	MAL SERR
1	High	Level	MAL TXDE
2	High	Level	MAL RXDE
3	High	Level	UART2
4	High	Level	UART3
5	High	Edge	External Bus Controller
6	High	Edge	NDFC Interrupt
7	High	Level	Kasumi Slave Error
8	High	Level	GPT Compare Timer 5
9	High	Level	GPT Compare Timer 6
10	High	Level	PLB3x4x MIRQ 0
11	High	Level	PLB3x4x MIRQ 1
12	High	Level	PLB3x4x MIRQ 2
13	High	Level	PLB3x4x MIRQ 3
14	High	Level	PLB3x4x MIRQ 4
15	High	Level	PLB3x4x MIRQ 5
16	High	Level	GPT Compare Timer 0
17	High	Level	GPT Compare Timer 1
18	Programmable	Programmable	External IRQ 7
19	Programmable	Programmable	External IRQ 8
20	Programmable	Programmable	External IRQ 9
21	High	Level	GPT Compare Timer 2
22	High	Level	GPT Compare Timer 3
23	High	Level	GPT Compare Timer 4
24	High	Level	Serial ROM
25	Low	Edge	GPT Decrement pulse
26			None
27	Low	Level	EXT_PCI_PERR (Parity)
28	Programmable	Programmable	External IRQ 0
29	High	Level	EMAC 0 Wake-up
30	Programmable	Programmable	External IRQ 1
31	High	Level	EMAC 1 Wake-up

Table 9-3. UIC2 Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	Programmable	Programmable	External IRQ 5
1	Programmable	Programmable	External IRQ 6
2	High	Level	OPB2PLB4 bridge *(PPC440EPX only)*
3	Programmable	Programmable	External IRQ 2
4	Programmable	Programmable	External IRQ 3
5	High	Level	DDR2 SDRAM controller
6	High	Edge	MAL Coalescence TX0
7	High	Edge	MAL Coalescence TX1
8	High	Edge	MAL Coalescence RX0
9	High	Edge	MAL Coalescence RX1
10:31			None

9.4 Interrupt Programmability

All of the on-chip interrupts and the external IRQs are programmable. However, the polarity and sensitivity of the on-chip interrupts must be programmed as shown in *Table 9-1*, *Table 9-2*, and *Table 9-3*, using the UIC Polarity Register (UIC*x*_PR) and UIC Trigger Register (UIC*x*_TR).

9.5 UIC Registers

The UIC is controlled through the Device Control Registers (DCRs) listed in *Table 9-4*. The registers are accessed using the **mfdcr** and **mtdcr** instructions.

Table 9-4. UIC Device Control Registers

Mnemonic	Register	Address	Access	Page
UIC0_SR	UIC Status Register 0	0x0C0	Read/Clear	210
UIC1_SR	UIC Status Register 1	0x0D0	Read/Clear	212
UIC2_SR	UIC Status Register 2	0x0E0	Read/Clear	215
UIC0_ER	UIC Enable Register 0	0x0C2	R/W	215
UIC1_ER	UIC Enable Register 1	0x0D2	R/W	218
UIC2_ER	UIC Enable Register 2	0x0E2	R/W	220
UIC0_CR	UIC Critical Register 0	0x0C3	R/W	221
UIC1_CR	UIC Critical Register 1	0x0D3	R/W	223
UIC2_CR	UIC Critical Register 2	0x0E3	R/W	225
UIC0_PR	UIC Polarity Register 0	0x0C4	R/W	226
UIC1_PR	UIC Polarity Register 1	0x0D4	R/W	229
UIC2_PR	UIC Polarity Register 2	0x0E4	R/W	231
UIC0_TR	UIC Trigger Register 0	0x0C5	R/W	232
UIC1_TR	UIC Trigger Register 1	0x0D5	R/W	234
UIC2_TR	UIC Trigger Register 2	0x0E5	R/W	237
UIC0_MSR	UIC Masked Status Register 0	0x0C6	Read-only	237
UIC1_MSR	UIC Masked Status Register 1	0x0D6	Read-only	240
UIC2_MSR	UIC Masked Status Register 2	0x0E6	Read-only	242
UIC0_VR	UIC Vector Register 0	0x0C7	Read-only	244
UIC1_VR	UIC Vector Register 1	0x0D7	Read-only	245
UIC0_VCR	UIC Vector Configuration Register 0	0x0C8	Write-only	243
UIC1_VCR	UIC Vector Configuration Register 1	0x0D8	Write-only	243

9.5.1 UIC0 Status Register (UIC0_SR)

To report interrupt status, the UIC0_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC0_SR fields.

Figure 9-2. UIC0 Status Register (UIC0_SR)

0	U0	UART0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
1	U1	UART1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	

2	IIC0	IIC0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
3	KRD	Kasumi Ready for Data Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
4	KDA	Kasumi Data Available Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
5	PCRW	PCI Command Register Write Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
6	PPM	PCI Power Management Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
7	IIC1	IIC1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
8	SPI	SPI Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
9	EPS	External PCI SERR Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
10	MTE	MAL TX EOB Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
11	MRE	MAL RX EOB Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
12	D0	DMA Channel 0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
13	D1	DMA Channel 1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
14	D2	DMA Channel 2 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
15	D3	DMA Channel 3 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
16	UDMA0	UDMA IRQ 0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
17	UDMA1	UDMA IRQ 1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
18	UDMA2	UDMA IRQ 2 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
19	UDMA3	UDMA IRQ 3 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	

20	USB2D	USB2.0 Device Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	*(PPC440EPX only)*
21	USB2H1	USB2.0 Host OHCI IRQ 1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	USB2 general interrupt from OHCI Host Controller.
22		Reserved	*(PPC440EPX only)*
23	SEIP	Security Function Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
24	EMAC0	EMAC 0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
25	EMAC1	EMAC 1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
26	USB2HE	USB2.0 Host EHCI Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	*(PPC440EPX only)*
27	EIR4	External IRQ 4 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
28	UIC2NC	UIC2 Non-Critical Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
29	UIC2C	UIC2 Critical Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
30	UIC1NC	UIC1 Non-Critical Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
31	UIC1C	UIC1 Critical Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	

9.5.2 UIC1 Status Register (UIC1_SR)

To report interrupt status, the UIC1_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC1_SR fields.

Figure 9-3. UIC1 Status Register (UIC1_SR)

0	MS	MAL SERR Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
1	MTDE	MAL TXDE Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	

Figure 9-3. UIC1 Status Register (UIC1_SR) (continued)			
2	MRDE	MAL RXDE Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
3	U2	UART2 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
4	U3	UART3 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
5	EBC	EBC Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
6	NDFC	NDFC Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
7	KSEI	Kasumi Slave Error Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
8	CT5	GPT Compare Timer 5 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
9	CT6	GPT Compare Timer 6 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
16	CT0	GPT Compare Timer 0 Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
17	CT1	GPT Compare Timer 1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	

Figure 9-3. UIC1 Status Register (UIC1_SR) (continued)

18	EIR7	External IRQ 7 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
19	EIR8	External IRQ 8 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
20	EIR9	External IRQ 9 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
21	CT2	GPT Compare Timer 2 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
22	CT3	GPT Compare Timer 3 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
23	CT4	GPT Compare Timer 4 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
24	SRE	Serial ROM Error Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
25	GDP	GPT Decrement Pulse Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
26		Reserved	
27	EPP	EXT_PCI_PERR (parity) Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	

9.5.3 UIC2 Status Register (UIC2_SR)

To report interrupt status, the UIC2_SR fields capture and hold internal and external interrupts until the fields are intentionally reset. To reset a field, write 1 to the field.

The values of other UIC registers do not affect UIC2_SR fields.

Figure 9-4. UIC2 Status Register (UIC2_SR)			
0	EIR5	External IRQ 5 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
1	EIR6	External IRQ 6 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
2	OPB	OPB to PLB4 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	*(PPC440EPX only)*
3	EIR2	External IRQ 2 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
4	EIR3	External IRQ 3 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
5	DDR2	DDR2 SDRAM Controller Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
6	MCT0	MAL Coalesence TX0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
7	MCT1	MAL Coalesence TX1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
8	MCR0	MAL Coalesence RX0 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
9	MCR1	MAL Coalesence RX1 Interrupt Status 0 Interrupt has not occurred. 1 Interrupt occurred.	
10:31		Reserved	

9.5.4 UIC0 Enable Register (UIC0_ER)

The fields of the UIC0_ER, which correspond to the fields of the UIC0_SR, enable or disable the reporting of the corresponding fields of the UIC0_SR.

If a UIC0_ER field is set to 1, the corresponding field of the UIC0_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC0_SR field is set to 1. If a UIC0_ER field is set to 0, the corresponding field of the UIC0_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC0_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC0_CR.

Figure 9-5. UIC0 Enable Register (UIC0_ER)

0	U0	UART0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
1	U1	UART1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
2	IIC0	IIC0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
3	KRD	Kasumi Ready for Data Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
4	KDA	Kasumi Data Available Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
5	PCRW	PCI Command Register Write Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
6	PPM	PCI Power Management Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
7	IIC1	IIC1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
8	SPI	SPI Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
9	EPS	External PCI SERR Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
10	MTE	MAL TX EOB Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
11	MRE	MAL RX EOB Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
12	D0	DMA Channel 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
13	D1	DMA Channel 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
14	D2	DMA Channel 2 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
15	D3	DMA Channel 3 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	

Figure 9-5. UIC0 Enable Register (UIC0_ER) (continued)			
16	UDMA0	UDMA IRQ 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
17	UDMA1	UDMA IRQ 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
18	UDMA2	UDMA IRQ 2 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
19	UDMA3	UDMA IRQ 3 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
20	USB2D	USB2.0 Device Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	*(PPC440EPX only)*
21	USB2H1	USB2.0 Host OHCI IRQ 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	USB2 general interrupt from OHCI Host Controller. *(PPC440EPX only)*
22		Reserved	
23	SEIP	Security Function Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
24	EMAC0	EMAC 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
25	EMAC1	EMAC 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
26	USB2HE	USB2.0 Host EHCI Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	*(PPC440EPX only)*
27	EIR4	External IRQ 4 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
28	UIC2NC	UIC2 Non-Critical Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
29	UIC2C	UIC2 Critical Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
30	UIC1NC	UIC1 Non-Critical Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
31	UIC1C	UIC1 Critical Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	

9.5.5 UIC1 Enable Register (UIC1_ER)

The fields of the UIC1_ER, which correspond to the fields of the UIC1_SR, enable or disable the reporting of the corresponding fields of the UIC1_SR.

If a UIC1_ER field is set to 1, the corresponding field of the UIC1_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC1_SR field is set to 1. If a UIC1_ER field is set to 0, the corresponding field of the UIC1_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC1_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC1_CR.

Figure 9-6. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
1	MTDE	MAL TXDE Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
2	MRDE	MAL RXDE Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
3	U2	UART2 Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
4	U3	UART3 Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
5	EBC	EBC Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
6	NDFC	NDFC Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
7	KSEI	Kasumi Slave Error Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
8	CT5	GPT Compare Timer 5 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
9	CT6	GPT Compare Timer 6 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	

Figure 9-6. UIC1 Enable Register (UIC1_ER) (continued)			
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
16	CT0	GPT Compare Timer 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
17	CT1	GPT Compare Timer 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
18	EIR7	External IRQ 7 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
19	EIR8	External IRQ 8 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
20	EIR9	External IRQ 9 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
21	CT2	GPT Compare Timer 2 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
22	CT3	GPT Compare Timer 3 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
23	CT4	GPT Compare Timer 4 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
24	SRE	Serial ROM Error Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
25	GDP	GPT Decrement Pulse Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
26		Reserved	
27	EPP	EXT_PCI_PERR (parity) Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
28	EIR0	External IRQ 0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	

Figure 9-6. UIC1 Enable Register (UIC1_ER) (continued)			
29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
30	EIR1	External IRQ 1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	

9.5.6 UIC2 Enable Register (UIC2_ER)

The fields of the UIC2_ER, which correspond to the fields of the UIC2_SR, enable or disable the reporting of the corresponding fields of the UIC2_SR.

If a UIC2_ER field is set to 1, the corresponding field of the UIC2_SR generates a critical or non-critical interrupt signal to the processor core, if the UIC2_SR field is set to 1. If a UIC2_ER field is set to 0, the corresponding field of the UIC2_SR does not generate a critical or non-critical interrupt signal to the processor core, regardless of the setting of the UIC2_SR field. The critical and non-critical interrupt signals in the processor core are controlled by fields in the Machine State Register (MSR).

The class of generated signals (critical or non-critical) is controlled by the UIC2_CR.

Figure 9-7. UIC2 Enable Register (UIC2_ER)			
0	EIR5	External IRQ 5 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
1	EIR6	External IRQ 6 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
2	OPB	OPB to PLB4 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	*(PPC440EPX only)*
3	EIR2	External IRQ 2 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
4	EIR3	External IRQ 3 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
5	DDR2	DDR2 SDRAM Controller Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
6	MCT0	MAL Coalesence TX0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	

Figure 9-7. UIC2 Enable Register (UIC2_ER)			
7	MCT1	MAL Coalesence TX1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
8	MCR0	MAL Coalesence RX0 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
9	MCR1	MAL Coalesence RX1 Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.	
10:31		Reserved	

9.5.7 UIC0 Critical Register (UIC0_CR)

The fields of the UIC0_CR, which correspond to the fields of the UIC0_SR and UIC0_ER, determine whether an interrupt captured in the corresponding fields of the UIC0_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC0_ER. The processor handles non-critical interrupts when MSR[EE] = 1 and critical interrupts when MSR[CE]=1.

If a UIC0_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC0_SR and enabled in the corresponding field of the UIC0_ER) generates a non-critical interrupt signal to the processor core. If a UIC0_CR field is a 1, a critical interrupt signal is generated.

Figure 9-8. UIC0 Critical Register (UIC0_CR)			
0	U0	UART0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
1	U1	UART1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
2	IIC0	IIC0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
3	KRD	Kasumi Ready for Data Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
4	KDA	Kasumi Data Available Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
5	PCRW	PCI Command Register Write Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
6	PPM	PCI Power Management Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
7	IIC1	IIC1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	

Figure 9-8. UIC0 Critical Register (UIC0_CR) (continued)

8	SPI	SPI Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
9	EPS	Ext PCI SERR Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
10	MTE	MAL TX EOB Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
11	MRE	MAL RX EOB Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
12	D0	DMA Channel 0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
13	D1	DMA Channel 1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
14	D2	DMA Channel 2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
15	D3	DMA Channel 3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
16	UDMA0	UDMA IRQ 0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
17	UDMA1	UDMA IRQ 1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
18	UDMA2	UDMA IRQ 2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
19	UDMA3	UDMA IRQ 3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
20	USB2D	USB2.0 Device Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	*(PPC440EPX only)*
21	USB2H1	USB2.0 Host OHCI IRQ 1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	USB2 general interrupt from OHCI Host Controller. *(PPC440EPX only)*
22		Reserved	
23	SEIP	Security Function Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
24	EMAC0	EMAC 0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	

Figure 9-8. UIC0 Critical Register (UIC0_CR) (continued)			
25	EMAC1	EMAC 1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
26	USB2HE	USB2.0 Host EHCI Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	*(PPC440EPX only)*
27	EIR4	External IRQ 4 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
28	UIC2NC	UIC2 Non-Critical Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
29	UIC2C	UIC2 Critical Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
30	UIC1NC	UIC1 Non-Critical Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
31	UIC1C	UIC1 Critical Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	

9.5.8 UIC1 Critical Register (UIC1_CR)

The fields of the UIC1_CR, which correspond to the fields of the UIC1_SR and UIC1_ER, determine whether an interrupt captured in the corresponding fields of the UIC1_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC1_ER. The processor handles non-critical interrupts when MSR[EE] = 1 and critical interrupts when MSR[CE]=1.

If a UIC1_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC1_SR and enabled in the corresponding field of the UIC1_ER) generates a non-critical interrupt signal to the processor core. If a UIC1_CR field is a 1, a critical interrupt signal is generated.

Figure 9-9. UIC1 Critical Register (UIC1_CR)			
0	MS	MAL SERR Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
1	MTDE	MAL TXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
2	MRDE	MAL RXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
3	U2	UART2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	

Figure 9-9. UIC1 Critical Register (UIC1_CR) (continued)

4	U3	UART3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
5	EBC	EBC Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
6	NDFC	NDFC Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
7	KSEI	Kasumi Slave Error Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
8	CT5	GPT Compare Timer 5 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
9	MS	MAL SERR Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
10	MTDE	MAL TXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
11	MRDE	MAL RXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
12	U2	UART2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
13	U3	UART3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
14	EBC	EBC Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
15	NDFC	NDFC Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
16	KSEI	Kasumi Slave Error Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
17	CT5	GPT Compare Timer 5 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
18	MS	MAL SERR Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
19	MTDE	MAL TXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	

Figure 9-9. UIC1 Critical Register (UIC1_CR) (continued)			
20	MRDE	MAL RXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
21	U2	UART2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
22	U3	UART3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
23	EBC	EBC Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
24	NDFC	NDFC Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
25	KSEI	Kasumi Slave Error Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
26	CT5	GPT Compare Timer 5 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
27	MS	MAL SERR Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
28	MTDE	MAL TXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
29	MRDE	MAL RXDE Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
30	U2	UART2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
31	U3	UART3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	

9.5.9 UIC2 Critical Register (UIC2_CR)

The fields of the UIC2_CR, which correspond to the fields of the UIC2_SR and UIC2_ER, determine whether an interrupt captured in the corresponding fields of the UIC2_SR generates a non-critical or critical interrupt, if the interrupts are enabled in the corresponding fields of the UIC2_ER. The processor handles non-critical interrupts when MSR[EE] = 1 and critical interrupts when MSR[CE]=1.

If a UIC2_CR field is set to 0, an enabled interrupt (captured in the corresponding field of the UIC2_SR and enabled in the corresponding field of the UIC2_ER) generates a non-critical interrupt signal to the processor core. If a UIC2_CR field is a 1, a critical interrupt signal is generated.

Figure 9-10. UIC2 Critical Register (UIC2_CR)			
0	EIR5	External IRQ 5 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
1	EIR6	External IRQ 6 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
2	OPB	OPB to PLB4 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	*(PPC440EPX only)*
3	EIR2	External IRQ 2 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
4	EIR3	External IRQ 3 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
5	DDR2	DDR2 SDRAM Controller Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
6	MCT0	MAL Coalesence TX0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
7	MCT1	MAL Coalesence TX1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
8	MCR0	MAL Coalesence RX0 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
9	MCR1	MAL Coalesence RX1 Interrupt Class 0 Interrupt is non-critical. 1 Interrupt is critical.	
10:31		Reserved	

9.5.10 UIC0 Polarity Register (UIC0_PR)

The fields of the UIC0_PR, which correspond to the fields of the UIC0_SR, determine whether the corresponding fields in the UIC0_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC0_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC0_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC0_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC0_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC0_PR) have positive polarity, the associated fields must be set to 1.

Figure 9-11. UIC0 Polarity Register (UIC0_PR)

0	U0	UART0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
1	U1	UART1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
2	IIC0	IIC0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
3	KRD	Kasumi Ready for Data Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
4	KDA	Kasumi Data Available Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
5	PCRW	PCI Command Register Write Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
6	PPM	PCI Power Management Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
7	IIC1	IIC1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
8	SPI	SPI Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
9	EPS	Ext PCI SERR Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 0
10	MTE	MAL TX EOB Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
11	MRE	MAL RX EOB Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
12	D0	DMA Channel 0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
13	D1	DMA Channel 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
14	D2	DMA Channel 2 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1

Figure 9-11. UIC0 Polarity Register (UIC0_PR) (continued)

15	D3	DMA Channel 3 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
16	UDMA0	UDMA IRQ 0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
17	UDMA1	UDMA IRQ 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
18	UDMA2	UDMA IRQ 2 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
19	UDMA3	UDMA IRQ 3 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
20	USB2D	USB2.0 Device Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 0 *(PPC440EPX only)*
21	USB2H1	USB2.0 Host OHCI IRQ 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	USB2 general interrupt from OHCI Host Controller. Must be set to 0 *(PPC440EPX only)*
22		Reserved	
23	SEIP	Security Function Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
24	EMAC0	EMAC 0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
25	EMAC1	EMAC 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
26	USB2HE	USB2.0 Host EHCI Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1 *(PPC440EPX only)*
27	EIR4	External IRQ 4 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
28	UIC2NC	UIC2 Non-Critical Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
29	UIC2C	UIC2 Critical Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
30	UIC1NC	UIC1 Non-Critical Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
31	UIC1C	UIC1 Critical Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1

9.5.11 UIC1 Polarity Register (UIC1_PR)

The fields of the UIC1_PR, which correspond to the fields of the UIC1_SR, determine whether the corresponding fields in the UIC1_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC1_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC1_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC1_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC1_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC1_PR) have positive polarity, the associated fields must be set to 1.

Figure 9-12. UIC1 Polarity Register (UIC1_PR)

0	MS	MAL SERR Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
1	MTDE	MAL TXDE Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
2	MRDE	MAL RXDE Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
3	U2	UART2 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
4	U3	UART3 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
5	EBC	EBC Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
6	NDFC	NDFC Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
7	KSEI	Kasumi Slave Error Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
8	CT5	GPT Compare Timer 5 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
9	CT6	GPT Compare Timer 6 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1

Figure 9-12. UIC1 Polarity Register (UIC1_PR) (continued)

11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
16	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
17	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
18	EIR7	External IRQ 7 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
19	EIR8	External IRQ 8 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
20	EIR9	External IRQ 9 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
21	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
22	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
23	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
24	SRE	Serial ROM Error Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
25	GDP	GPT Decrement Pulse Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 0
26		Reserved	
27	EPP	EXT_PCI_PERR (parity) Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 0

Figure 9-12. UIC1 Polarity Register (UIC1_PR) (continued)			
28	EIR0	External IRQ 0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
30	EIR1	External IRQ 1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1

9.5.12 UIC2 Polarity Register (UIC2_PR)

The fields of the UIC2_PR, which correspond to the fields of the UIC2_SR, determine whether the corresponding fields in the UIC2_SR have a positive or negative polarity.

For level-sensitive interrupts, a 0 in a UIC2_PR field causes the corresponding interrupt to be negative active. A 1 in a UIC2_PR field causes the corresponding interrupt to be positive active.

For edge-sensitive interrupts, a 0 in a UIC2_PR field causes the corresponding interrupt to be detected on a falling edge (as polarity changes from 1 to 0). A 1 in a UIC2_PR field causes the corresponding interrupt to be detected on a rising edge (as polarity changes from 0 to 1).

Because the on-chip interrupts (those controlled by UIC2_PR) have positive polarity, the associated fields must be set to 1.

Figure 9-13. UIC2 Polarity Register (UIC2_PR)			
0	EIR5	External IRQ 5 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
1	EIR6	External IRQ 6 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
2	OPB	OPB to PLB4 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1 *(PPC440EPX only)*
3	EIR2	External IRQ 2 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
4	EIR3	External IRQ 3 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	
5	DDR2	DDR2 SDRAM Controller Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1

Figure 9-13. UIC2 Polarity Register (UIC2_PR)			
6	MCT0	MAL Coalesence TX0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
7	MCT1	MAL Coalesence TX1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
8	MCR0	MAL Coalesence RX0 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
9	MCR1	MAL Coalesence RX1 Interrupt Polarity 0 Interrupt has negative polarity. 1 Interrupt has positive polarity.	Must be set to 1
10:31		Reserved	

9.5.13 UIC0 Trigger Register (UIC0_TR)

The fields of the UIC0_TR, which correspond to the fields of the UIC0_SR, determine whether corresponding fields in the UIC0_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC0_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC0_TR field is 0, the associated interrupt is level-sensitive. If the UIC0_TR field is 1, the interrupt is edge-sensitive.

Figure 9-14. UIC0 Trigger Register (UIC0_TR)			
0	U0	UART0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
1	U1	UART1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
2	IIC0	IIC0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
3	KRD	Kasumi Ready for Data Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
4	KDA	Kasumi Data Available Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0

Figure 9-14. UIC0 Trigger Register (UIC0_TR) (continued)			
5	PCRW	PCI Command Register Write Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
6	PPM	PCI Power Management Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
7	IIC1	IIC1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
8	SPI	SPI Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
9	EPS	Ext PCI SERR Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
10	MTE	MAL TX EOB Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
11	MRE	MAL RX EOB Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
12	D0	DMA Channel 0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
13	D1	DMA Channel 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
14	D2	DMA Channel 2 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
15	D3	DMA Channel 3 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
16	UDMA0	UDMA IRQ 0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
17	UDMA1	UDMA IRQ 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
18	UDMA2	UDMA IRQ 2 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
19	UDMA3	UDMA IRQ 3 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
20	USB2D	USB2.0 Device Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1 *(PPC440EPX only)*

Figure 9-14. UIC0 Trigger Register (UIC0_TR) (continued)			
21	USB2H1	USB2.0 Host OHCI IRQ 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	USB2 general interrupt from OHCI Host Controller. Must be set to 0 *(PPC440EPX only)*
22		Reserved	
23	SEIP	Security Function Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
24	EMAC0	EMAC 0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
25	EMAC1	EMAC 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
26	USB2HE	USB2.0 Host EHCI Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0 *(PPC440EPX only)*
27	EIR4	External IRQ 4 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
28	UIC2NC	UIC2 Non-Critical Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
29	UIC2C	UIC2 Critical Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
30	UIC1NC	UIC1 Non-Critical Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
31	UIC1C	UIC1 Critical Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0

9.5.14 UIC1 Trigger Register (UIC1_TR)

The fields of the UIC1_TR, which correspond to the fields of the UIC1_SR, determine whether corresponding fields in the UIC1_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC1_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC1_TR field is 0, the associated interrupt is level-sensitive. If the UIC1_TR field is 1, the interrupt is edge-sensitive.

Figure 9-15. UIC1 Trigger Register (UIC1_TR)			
0	MS	MAL SERR Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
1	MTDE	MAL TXDE Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
2	MRDE	MAL RXDE Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
3	U2	UART2 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
4	U3	UART3 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
5	EBC	EBC Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
6	NDFC	NDFC Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
7	KSEI	Kasumi Slave Error Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
8	CT5	GPT Compare Timer 5 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
9	CT6	GPT Compare Timer 6 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
10	P2P0	PLB3 to PLB4 Bridge 0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
11	P2P1	PLB3 to PLB4 Bridge 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
12	P2P2	PLB3 to PLB4 Bridge 2 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
13	P2P3	PLB3 to PLB4 Bridge 3 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
14	P2P4	PLB3 to PLB4 Bridge 4 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
15	P2P5	PLB3 to PLB4 Bridge 5 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0

Figure 9-15. UIC1 Trigger Register (UIC1_TR) (continued)

16	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
17	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
18	EIR7	External IRQ 7 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
19	EIR8	External IRQ 8 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
20	EIR9	External IRQ 9 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
21	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
22	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
23	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
24	SRE	Serial ROM Error Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
25	GDP	GPT Decrement Pulse Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
26		Reserved	
27	EPP	EXT_PCI_PERR (parity) Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
28	EIR0	External IRQ 0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
30	EIR1	External IRQ 1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0

9.5.15 UIC2 Trigger Register (UIC2_TR)

The fields of the UIC2_TR, which correspond to the fields of the UIC2_SR, determine whether corresponding fields in the UIC2_SR are edge-sensitive or level-sensitive.

Edge-sensitive interrupts are triggered depending on whether the associated interrupt signal is rising or falling (changing from 0 to 1 or 1 to 0, respectively). Whether a rising or falling edge causes the trigger is controlled by bits in the UIC2_PR.

Level-sensitive interrupts are triggered depending on whether the associated interrupt signal is high (1) or low (0).

If a UIC2_TR field is 0, the associated interrupt is level-sensitive. If the UIC1_TR field is 1, the interrupt is edge-sensitive.

Figure 9-16. UIC2 Trigger Register (UIC2_TR)

0	EIR5	External IRQ 5 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
1	EIR6	External IRQ 6 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
2	OPB	OPB to PLB4 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0 *(PPC440EPX only)*
3	EIR2	External IRQ 2 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
4	EIR3	External IRQ 3 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	
5	DDR2	DDR2 SDRAM Controller Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 0
6	MCT0	MAL Coalesence TX0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
7	MCT1	MAL Coalesence TX1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
8	MCR0	MAL Coalesence RX0 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
9	MCR1	MAL Coalesence RX1 Interrupt Trigger 0 Interrupt is level sensitive. 1 Interrupt is edge sensitive.	Must be set to 1
10:31		Reserved	

9.5.16 UIC0 Masked Status Register (UIC0_MSR)

This read-only register contains the result of masking the UIC0_SR with the UIC0_ER. Reading this register, instead of the actual UIC0_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC0_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC0_SR, the UIC0_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC0_SR can be successfully cleared.

Figure 9-17. UIC0 Masked Status Register (UIC0_MSR)

0	U0	UART0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
1	U1	UART1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
2	IIC0	IIC0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
3	KRD	Kasumi Ready for Data Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
4	KDA	Kasumi Data Available Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
5	PCRW	PCI Command Register Write Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
6	PPM	PCI Power Management Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
7	IIC1	IIC1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
8	SPI	SPI Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
9	EPS	Ext PCI SERR Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
10	MTE	MAL TX EOB Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
11	MRE	MAL RX EOB Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
12	D0	DMA2P30 Channel 0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
13	D1	DMA2P30 Channel 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
14	D2	DMA2P30 Channel 2 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	

Figure 9-17. UIC0 Masked Status Register (UIC0_MSR) (continued)			
15	D3	DMA2P30 Channel 3 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
16	UDMA0	UDMA IRQ 0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
17	UDMA1	UDMA IRQ 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
18	UDMA2	UDMA IRQ 2 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
19	UDMA3	UDMA IRQ 3 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
20	USB2D	USB2.0 Device Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	*(PPC440EPX only)*
21	USB2H1	USB2.0 Host OHCI IRQ 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	USB2 general interrupt from OHCI Host Controller. *(PPC440EPX only)*
22		Reserved	
23	SEIP	Security Function Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
24	EMAC0	EMAC 0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
25	EMAC1	EMAC 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
26	USB2HE	USB2.0 Host EHCI Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	*(PPC440EPX only)*
27	EIR4	External IRQ 4 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
28	UIC2NC	UIC2 Non-Critical Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
29	UIC2C	UIC2 Critical Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
30	UIC1NC	UIC1 Non-Critical Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
31	UIC1C	UIC1 Critical Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	

9.5.17 UIC1 Masked Status Register (UIC1_MSR)

This read-only register contains the result of masking the UIC1_SR with the UIC1_ER. Reading this register, instead of the actual UIC1_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC1_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC1_SR, the UIC1_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC1_SR can be successfully cleared.

Figure 9-18. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
1	MTDE	MAL TXDE Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
2	MRDE	MAL RXDE Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
3	U2	UART2 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
4	U3	UART3 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
5	EBC	EBC Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
6	NDFC	NDFC Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
7	KSEI	Kasumi Slave Error Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
8	CT5	GPT Compare Timer 5 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
9	CT6	GPT Compare Timer 6 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
10	P2P0	PLB3 to PLB4 Bridge 0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
11	P2P1	PLB3 to PLB4 Bridge 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
12	P2P2	PLB3 to PLB4 Bridge 2 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	

Figure 9-18. UIC1 Masked Status Register (UIC1_MSR) (continued)			
13	P2P3	PLB3 to PLB4 Bridge 3 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
14	P2P4	PLB3 to PLB4 Bridge 4 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
15	P2P5	PLB3 to PLB4 Bridge 5 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
16	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
17	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
18	EIR7	External IRQ 7 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
19	EIR8	External IRQ 8 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
20	EIR9	External IRQ 9 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
21	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
22	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
23	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
24	SRE	Serial ROM Error Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
25	GDP	GPT Decrement Pulse Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
26		Reserved	
27	EPP	EXT_PCI_PERR (parity) Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
28	EIR0	External IRQ 0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	

Figure 9-18. UIC1 Masked Status Register (UIC1_MSR) (continued)			
30	EIR1	External IRQ 1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	

9.5.18 UIC2 Masked Status Register (UIC2_MSR)

This read-only register contains the result of masking the UIC2_SR with the UIC2_ER. Reading this register, instead of the actual UIC1_SR, eliminates the need for software to read and apply the enable mask to the contents of the UIC2_SR to determine which enabled interrupt fields are active.

If an interrupt is configured as level-sensitive, and a clear is attempted on the UIC2_SR, the UIC2_SR field is not cleared if the incoming interrupt signal is at the asserted polarity. The interrupt signal must be reset before the UIC2_SR can be successfully cleared.

Figure 9-19. UIC2 Maked Status Register (UIC2_MSR)			
0	EIR5	External IRQ 5 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
1	EIR6	External IRQ 6 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
2	OPB	OPB to PLB4 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	*(PPC440EPX only)*
3	EIR2	External IRQ 2 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
4	EIR3	External IRQ 3 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
5	DDR2	DDR2 SDRAM Controller Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
6	MCT0	MAL Coalesence TX0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
7	MCT1	MAL Coalesence TX1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
8	MCR0	MAL Coalesence RX0 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
9	MCR1	MAL Coalesence RX1 Masked Interrupt Status 0 Masked interrupt has not occurred. 1 Masked interrupt occurred.	
10:31		Reserved	

9.5.19 UIC0 Vector Configuration Register (UIC0_VCR)

The write-only UIC0_VCR enables software control of interrupt vector generation for critical interrupts. UIC0_VCR contains an address, used as an interrupt vector base address, and specifies interrupt ordering priority. Vector generation is not performed for non-critical interrupts.

UIC0_VCR[VBA] can contain either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC0_VR, using UIC0_VCR[VBA]. Vector generation is described in *UIC0 Vector Register (UIC0_VR)* on page 244. Because the two lowest-order bits of an interrupt handler address are assumed to be 00 to ensure word alignment, 30 bits are sufficient to form the base address.

A general interrupt handler uses the vector to access a table of interrupt vectors. Each interrupt vector table entry contains the address of an interrupt handler for a specific interrupt. Alternatively, UIC0_VCR[VBA] can directly address the interrupt handlers for specific interrupts, which in memory are separated by an offset calculated in UIC0_VR.

UIC0_VCR[PRO] controls whether the interrupt associated with UIC0_SR[0] or UIC0_SR[31] has the highest priority. If UIC0_VCR[PRO] = 0, the interrupt associated with UIC0_SR[31] has the highest priority; if UIC0_VCR[PRO] = 1, the interrupt associated with UIC0_SR[0] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC0_CR as a interrupt has the second highest priority. Priority decreases across the UIC0_SR to the end opposite the highest priority field.

Figure 9-20. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address	
30		Reserved	
31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt.	**Note:** Vector generation is not performed for non-critical interrupts.

9.5.20 UIC1 Vector Configuration Register (UIC1_VCR)

The write-only UIC1_VCR enables software control of interrupt vector generation for critical interrupts. UIC1_VCR contains an address, used as an interrupt vector base address, and specifies interrupt ordering priority. Vector generation is not performed for non-critical interrupts.

UIC1_VCR[VBA] can contain either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC1_VR, using UIC1_VCR[VBA]. Vector generation is described in *UIC1 Vector Register (UIC1_VR)* on page 245. Because the two lowest-order bits of an interrupt handler address are assumed to be 00 to ensure word alignment, 30 bits are sufficient to form the base address.

A general interrupt handler uses the vector to access a table of interrupt vectors. Each interrupt vector table entry contains the address of an interrupt handler for a specific interrupt. Alternatively, UIC1_VCR[VBA] can directly address the interrupt handlers for specific interrupts, which in memory are separated by an offset calculated in UIC1_VR.

UIC1_VCR[PRO] controls whether the interrupt associated with UIC1_SR[0] or UIC1_SR[31] has the highest priority. If UIC1_VCR[PRO] = 0, the interrupt associated with UIC1_SR[31] has the highest priority; if UIC1_VCR[PRO] = 1, the interrupt associated with UIC1_SR[0] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC*x*_CR as a interrupt has the second highest priority. Priority decreases across the UIC1_SR to the end opposite the highest priority field.

Figure 9-21. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address	
30		Reserved	
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt.	**Note:** Vector generation is not performed for non-critical interrupts.

9.5.21 UIC0 Vector Register (UIC0_VR)

The read-only UIC0_VR contains an interrupt vector that can reduce interrupt handling latency for critical interrupts. Vector generation logic adds an offset to UIC0_VCR[VBA], and the sum is returned in the UIC0_VR. Vectors are not computed for non-critical interrupts.

The interrupt vector is based on the field position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC0_SR. The generated vectors can be programmed to point directly to the interrupt handlers.

> **Programming Note:** Regardless of the programming of UIC0_VCR and UIC0_VR registers, the processor always vectors to IVPR and IVOR0 when a critical interrupt occurs.

The interrupt vector offset is based on the bit position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC0_SR. The offset has a fixed value of 512 per bit. The main critical interrupt handler can interpret the vector returned by UIC0_VR as the address of the interrupt handler for that interrupt, assuming the routine is 512 bytes or smaller. Alternatively, the main critical interrupt handler can interpret the vector as a look-up table entry for the address of the interrupt handler for that interrupt.

Figure 9-22. UIC Vector Register (UIC0_VR)

0:31	VR	Interrupt Vector	

The following example illustrates the generation of a UIC0_VR vector for external interrupt request IRQ2.

For the example, assume that UIC0_VCR[PRO] = 0, so that UIC0_SR[EIR6S] (UIC0_SR_{31})has the highest interrupt priority, and that UIC0_SR[EIR2S] (UIC0_SR_{27}) is the current highest priority, enabled, active, critical interrupt. To generate the vector for the interrupt associated with UIC0_SR[EIR2S], internal logic multiplies the difference between the highest priority interrupt bit and the active enabled priority interrupt bit by 512. The interrupt vector offset is therefore $(31 - 27) \times 512 = 4 \times 512$. This offset is added to the base address in UIC0_VCR[VBA], and the UIC0_VR returns UIC0_VCR[VBA] + (4×512).

9.5.21.1 Using the Value in UIC0_VR as a Vector Address or Entry Table Lookup

If an interrupt handler is 512 bytes or smaller, system software can interpret the value returned in the UIC0_VR as an address. In this case, when the interrupt is received, the UIC0_VR is read and software simply jumps to the address represented by the UIC0_VR value. Alternatively, the routine can be at a different address, and system

software can treat the value of the UIC0_VR as a pointer, storing the interrupt handler address in the UIC0_VR during system initialization. In this case, when the interrupt is handled, software must read the UIC0_VR, read the entry at the UIC0_VR value, and jump to the entry. Hardware has no knowledge of the method is used, which is determined by system software.

9.5.21.2 Vector Generation Scenarios

For the following sequence, assume that the interrupts are enabled and critical (vectors are not generated for disabled or non-critical interrupts). The sequence illustrates several scenarios for vector generation.

1. An intermediate priority interrupt goes active; its vector is stored in UIC0_VR.
2. A low priority interrupt goes active; UIC0_VR is unchanged.
3. Software reads the vector; UIC0_VR is unchanged.
4. Software resets the intermediate priority interrupt; UIC0_VR contains the vector for the low priority interrupt.
5. A high priority interrupt goes active; UIC0_VR contains the vector for the high priority interrupt.
6. Software resets the high priority interrupt; UIC0_VR contains the vector for the low priority interrupt.
7. Software resets the UIC0_ER field for the low priority interrupt, disabling it; UIC0_VR contains 0x00000000.
8. UIC0_CR is reprogrammed to make the low priority interrupt non-critical and UIC0_ER is reprogrammed to re-enable the low priority interrupt; UIC0_VR continues to contain 0x00000000.

9.5.22 UIC1 Vector Register (UIC1_VR)

The read-only UIC1_VR contains an interrupt vector that can reduce interrupt handling latency for critical interrupts. Vector generation logic adds an offset to UIC1_VCR[VBA], and the sum is returned in the UIC1_VR. Vectors are not computed for non-critical interrupts.

The interrupt vector is based on the field position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC1_SR. The generated vectors can be programmed to point directly to the interrupt handlers.

> **Programming Note:** Regardless of the programming of UIC1_VCR and UIC1_VR registers, the processor always vectors to IVPR and IVOR0 when a critical interrupt occurs.

The interrupt vector offset is based on the bit position of the current highest priority, enabled, active, critical interrupt relative to the highest priority interrupt in the UIC1_SR. The offset has a fixed value of 512 per bit. The main critical interrupt handler can interpret the vector returned by UIC1_VR as the address of the interrupt handler for that interrupt, assuming the routine is 512 bytes or smaller. Alternatively, the main critical interrupt handler can interpret the vector as a look-up table entry for the address of the interrupt handler for that interrupt.

Figure 9-23. UIC1 Vector Register (UIC1_VR)

0:31		Interrupt Vector	

The following example illustrates the generation of a UIC1_VR vector for external interrupt request IRQ2.

For the example, assume that UIC1_VCR[PRO] = 0, so that UIC1_SR[EIR6S] ($UIC1\_SR_{31}$)has the highest interrupt priority, and that UIC1_SR[EIR2S] ($UIC1\_SR_{27}$) is the current highest priority, enabled, active, critical interrupt. To generate the vector for the interrupt associated with UIC1_SR[EIR2S], internal logic multiplies the

difference between the highest priority interrupt bit and the active enabled priority interrupt bit by 512. The interrupt vector offset is therefore (31 – 27) × 512 = 4 × 512. This offset is added to the base address in UIC1_VCR[VBA], and the UIC1_VR returns UIC1_VCR[VBA] + (4 × 512).

9.5.22.1 Using the Value in UIC1_VR as a Vector Address or Entry Table Lookup

If an interrupt handler is 512 bytes or smaller, system software can interpret the value returned in the UIC1_VR as an address. In this case, when the interrupt is received, the UIC1_VR is read and software simply jumps to the address represented by the UIC1_VR value. Alternatively, the routine can be at a different address, and system software can treat the value of the UIC1_VR as a pointer, storing the interrupt handler address in the UIC1_VR during system initialization. In this case, when the interrupt is handled, software must read the UIC1_VR, read the entry at the UIC1_VR value, and jump to the entry. Hardware has no knowledge of the method is used, which is determined by system software.

9.5.22.2 Vector Generation Scenarios

For the following sequence, assume that the interrupts are enabled and critical (vectors are not generated for disabled or non-critical interrupts). The sequence illustrates several scenarios for vector generation.

1. An intermediate priority interrupt goes active; its vector is stored in UIC1_VR.
2. A low priority interrupt goes active; UIC1_VR is unchanged.
3. Software reads the vector; UIC1_VR is unchanged.
4. Software resets the intermediate priority interrupt; UIC1_VR contains the vector for the low priority interrupt.
5. A high priority interrupt goes active; UIC1_VR contains the vector for the high priority interrupt.
6. Software resets the high priority interrupt; UIC1_VR contains the vector for the low priority interrupt.
7. Software resets the UIC1_ER field for the low priority interrupt, disabling it; UIC1_VR contains 0x00000000.
8. UIC1_CR is reprogrammed to make the low priority interrupt non-critical and UIC1_ER is reprogrammed to re-enable the low priority interrupt; UIC1_VR continues to contain 0x00000000.

10. Interrupts and Exceptions

Interrupt and exception processing for all chip functions except the FPU is handled by the PPC440 processor. Refer to the *PPC440 Processor User's Manual* for details.

Note: The FPU is supported by the PPC440EPx only.

11. Floating Point Unit Interrupts and Exceptions

An *interrupt* is the action in which the processor saves its old context (Machine State Register (MSR) and next instruction address NIA)) and begins execution at a pre-determined interrupt-handler address, with a modified MSR. *Exceptions* are the events that may cause the processor to take an interrupt, if the corresponding interrupt type is enabled.

Exceptions may be generated by the execution of instructions, or by signals from devices external to the PPC440 processor core, the internal timer facilities, debug events, or error conditions.

Note: The FPU is supported by the PPC440EPx only.

11.1 Floating-Point Exceptions

Book-E requires all synchronous (precise and imprecise) interrupts to be reported in program order, as required by the sequential execution model. The only exception to this rule is the case of multiple synchronous imprecise interrupts. Upon a synchronizing event, all previously executed instructions are required to report any synchronous imprecise interrupt-generating exceptions, and the interrupt is then generated with all of those exception types reported cumulatively, in both the Exception Syndrome Register (ESR), and any status registers associated with the particular exception type, such as the Floating-Point Status and Control Register (FPSCR).

For any single instruction attempting to cause multiple exceptions for which the corresponding synchronous interrupt types are enabled, this section defines the priority order by which the instruction will be permitted to cause a *single* enabled exception, thus generating a particular synchronous interrupt. This exception priority mechanism, along with the requirement that synchronous interrupts must be generated in program order, guarantees that only one of the synchronous interrupt types is considered at any given time. The exception priority mechanism also prevents certain debug exceptions from existing in combination with certain other synchronous interrupt-generating exceptions.

This section does not define the permitted setting of multiple exceptions for which the corresponding interrupt types are disabled. The generation of exceptions for which the corresponding interrupt types are disabled has no effect on the generation of other exceptions for which the corresponding interrupt types are enabled. Conversely, if a particular exception for which the corresponding interrupt type is enabled is shown in the following sections to be of a higher priority than another exception, it will prevent the setting of that other exception, regardless of whether the corresponding interrupt type of the other exception is enabled or disabled.

Except as noted, only one of the exception types listed for a given instruction type can be generated at any given time. The priority of the exception types are listed in subsequent sections ranging from highest to lowest, within each instruction type.

Note: Some exception types may be mutually exclusive of each other and could otherwise be considered the same priority. In such cases, the exceptions are listed in the order suggested by the sequential execution model.

Computational instructions may cause exceptions. Aside from instructions that write the FPSCR, none of the noncomputational instructions can cause a floating-point exception.

All exceptions are handled precisely. Because this can affect performance adversely, it is strongly recommended that exceptions should be disabled when possible. This prevents the PPC440 FPU instruction stream from waiting for the execution of long latency instructions, such as **fdiv**[**s**].

11.2 Exceptions List

Book-E defines the following floating-point exceptions:

Table 11-1. Invalid Operation Exception Categories

Category	FPSCR Field
SNaN	VXSNAN
Infinity – Infinity	VXISI
Infinity ÷ Infinity	VXIDI
Zero ÷ Zero	VXZDZ
Infinity × Zero	VXIMZ
Invalid Compare	VXVC
Software Request	VXSOFT
Invalid Square Root	VXSQRT
Invalid Integer Convert	VXCVI

- Invalid Operation exception (VX)
- Zero Divide exception (ZX)
- Overflow exception (OX)
- Underflow exception (UX)
- Inexact exception (XI)

These exceptions can occur during execution of computational instructions. In addition, an Invalid Operation exception occurs when a **mtsfs** or **mtsfsi** instruction sets FPSCR[VXSOFT] = 1.

Each floating-point exception, and each category of Invalid Operation exception, has an exception bit in the FPSCR. Each floating-point exception also has a corresponding enable bit in the FPSCR. The exception bit indicates the occurrence of the corresponding exception. If an exception occurs, the corresponding enable bit controls the result produced by the instruction and, with MSR[FE0, FE1] whether and how the Enabled exception type Program interrupt is taken. (See *Floating-Point Exceptions* on page 249 for more information.) In general, the enabling specified by an enable bit is to enable the invoking the interrupt, not to enable the exception to occur. The occurrence of an exception depends only on the instruction and its inputs, not on the setting of any enable bits. The only exceptions to this general rule are the occurrence of an Underflow exception, which may depend on the setting of the enable bit, and the occurrence of a Inexact exception, which may depend on the Overflow exception bit not being set.

A single instruction, other than **mtfsf** or **mtfsfi**, can set more than one exception bit only in the following cases:

- An Inexact exception may be set with an Overflow exception.
- An Inexact Exception may be set with an Underflow exception.
- An Invalid Operation exception (SNaN) is set with Invalid Operation exception ($\infty \times 0$) for *Multiply-Add* instructions for which the values being multiplied are infinity and 0, and the value being added is an SNaN.
- An Invalid Operation exception (SNaN) can be set with Invalid Operation exception (Invalid Compare) for *Compare Ordered* instructions.
- Invalid Operation exception (SNaN) can be set with Invalid Operation exception (Invalid Integer Convert) for *Convert To Integer* instructions.

When an exception occurs, instruction execution may be suppressed or a result may be delivered, depending on the exception.

Instruction execution is suppressed for the following kinds of exception, so that there is no possibility that one of the operands is lost:

- Enabled Invalid Operation
- Enabled Zero Divide

For the remaining exceptions, a result is generated and written to the target specified by the instruction causing the exception. The result may be a different value for the enabled and disabled conditions for some of these exceptions. The exceptions that deliver a result are:

- Disabled Invalid Operation
- Disabled Zero Divide
- Disabled Overflow
- Disabled Underflow
- Disabled Inexact
- Enabled Overflow
- Enabled Underflow
- Enabled Inexact

Subsequent sections define each of the floating-point exceptions and specify the action that is taken when they are detected.

IEEE 754 specifies the handling of exceptional conditions in terms of "traps" and "trap handlers." In Book-E, an FPSCR exception enable bit of 1 causes generation of the result value specified in the IEEE standard for the 'trap enabled' case. The exception is expected to be detected by software, which revises the result. An FPSCR exception enable bit of 0 causes generation of the "default result" value specified for the "trap disabled" (or "no trap occurs" or "trap is not implemented") case. Software is not expected to detect the exception, and simply uses the default result. The result to be delivered in each case for each exception is described in subsequent sections.

The IEEE 754 default behavior when an exception occurs is to generate a default value and to not notify software. In Book-E, if the IEEE 754 default behavior is desired for all exceptions, all FPSCR exception enable bits should be set to 0 and Ignore Exceptions Mode should be used (see *Table 11-2* on page 252). In this case, an Enabled exception type Program interrupt is not taken, even if floating-point exceptions occur. Software can inspect the FPSCR exception bits, if necessary, to determine whether exceptions have occurred.

If software is to be notified that a given kind of exception has occurred, the corresponding FPSCR exception enable bit must be set to 1 and a mode other than Ignore Exceptions Mode must be used. In this case, the Enabled exception type Program interrupt is taken if an enabled floating-point exception occurs. An Enabled exception type Program interrupt is also taken if an **mtsfs** or **mtsfsi** instruction sets an exception bit and its corresponding enable bit both to 1; the **mtsfs** or **mtsfsi** instruction is considered to cause the enabled exception.

MSR[FE0, FE1] control whether and how Enabled exception type Program interrupt are taken when an enabled floating-point exception occurs. An Enabled exception type Program interrupt is never taken because of a disabled floating-point exception.

Table 11-2. MSR[FE0, FE1] Modes

MSR[FE0]	MSR[FE1]	Mode
0	0	**Ignore Exceptions Mode** Floating-point exceptions do not cause an Enabled exception type Program interrupt to be taken.
1	1	**Precise Mode** An Enabled exception type Program interrupt is taken precisely at the instruction that caused the enabled exception.

If either MSR[FE0] or MSR[FE1] is 1, Enabled exception type Program interrupts are treated as in Precise Mode.

In all cases, the question of whether a floating-point result is stored, and what value is stored, is governed by the FPSCR exception enable bits, as described in subsequent sections, and is not affected by the value of MSR[FE0, FE1].

In all cases in which an Enabled exception type Program interrupt is taken, all instructions before the instruction at which the Enabled exception type Program interrupt is taken have completed, and no instruction after the instruction at which the Enabled exception type Program interrupt is taken has begun execution. The instruction at which the Enabled exception type Program interrupt is taken has not been executed unless it is the excepting instruction, in which case it has been executed if the exception is not an Enabled Invalid Operation exception or Enabled Zero Divide exception.

A **sync** instruction, or any other execution-synchronizing instruction or event, such as **isync**, also has the effects described above.

In order to obtain the best performance across the widest range of implementations, the programmer should follow these guidelines.

- If the IEEE 754 default results are acceptable to the application, Ignore Exceptions Mode should be used, with all FPSCR exception enable bits set to 0.
- Ignore Exceptions Mode should not, in general, be used when any FPSCR exception enable bits are set to 1.
- Precise Mode may degrade performance in some implementations, perhaps substantially, and therefore should be used only for debugging and other specialized applications.

11.3 Floating-Point Interrupts

The following interrupts are taken under the control of the PPC440 processor core, and are not enabled by or reported in FPSCR bits:

- Floating-Point Unavailable
- Floating-Point Assist

11.3.1 Floating-Point Unavailable Interrupt

A Floating-Point Unavailable interrupt occurs when no higher priority exception exists, an attempt is made to execute a floating-point instruction (including floating-point loads, stores, and moves), and MSR[FP] = 0.

When a Floating-Point Unavailable interrupt occurs, the processor suppresses the execution of the instruction causing the Floating-Point Unavailable interrupt.

11.4 Floating-Point Exception Behavior

The following sections describe the behavior that results from the floating-point exceptions. For each exception, the definition of the exception is given, followed by a description of the action caused by the exception.

In general, each exception can result in either of two types of action, depending on whether the exception is enabled by its associated exception enable bit in the FPSCR.

11.4.1 Invalid Operation Exception

An Invalid Operation exception occurs when an operand is invalid for the specified operation. The invalid operations are:

- Any floating-point operation on a signaling NaN (SNaN)
- For add or subtract operations, magnitude subtraction of infinities ($\infty - \infty$)
- Division of infinity by infinity ($\infty \div \infty$)
- Division of zero by zero ($0 \div 0$)
- Multiplication of infinity by zero ($\infty \times 0$)
- Ordered comparison involving a NaN (Invalid Compare)
- Square root or reciprocal square root of a negative and nonzero number (Invalid Square Root)
- Integer conversion involving a number too large in magnitude to be represented in the target format, or involving an infinity or a NaN (Invalid Integer Convert)

In addition, an Invalid Operation exception occurs if software explicitly requests this by executing an **mtfsf**, **mtfsfi**, or **mtfsb1** instruction that sets FPSCR[VXSOFT] = 1.

Programming Note: The purpose of FPSCR[VXSOFT] is to enable software to cause an Invalid Operation exception for a condition that is not necessarily associated with the execution of a floating-point instruction. For example, it may be set by a program that computes a square root, if the source operand is negative.

11.4.1.1 Action

The action taken depends on the setting of FPSCR[VE].

When Invalid Operation exception is enabled (FPSCR[VE] = 1) and an Invalid Operation exception occurs or software explicitly requests the exception, the following actions are taken:

- One or two FPSCR Invalid Operation exception bits, listed in *Table 11-3*, are set.

Table 11-3. Invalid Operation Exceptions

FPSCR Bit	Category
VXSNAN	SNaN
VXISI	Infinity – Infinity
VXIDI	Infinity ÷ Infinity
VXZDZ	Zero ÷ Zero
VXIMZ	Infinity × Zero
VXVC	Invalid Compare
VXSOFT	Software Request
VXSQRT	Invalid Square Root
VXCVI	Invalid Integer Convert

- If the operation is an arithmetic, **frsp**, or convert to integer operation, the target FPR is unchanged.
  - FPSCR[FR, FI] ← 0
  - FPSCR[FPRF] ← unchanged
- If the operation is a compare:
  - FPSCR[FR, FI, C] ← unchanged
  - FPSCR[FPCC] ← unordered
- If software explicitly requests the exception:
  - FPSCR[FR, FI, FPRF] are as set by the **mtfsf**, **mtfsfi**, or **mtfsb1** instruction.

When Invalid Operation exception is disabled (FPSCR[VE] = 0) and an Invalid Operation exception occurs, or software explicitly requests the exception, the following actions are taken:

- One or two FPSCR Invalid Operation exception bits, listed in *Table 11-3*, are set.
- If the operation is an arithmetic or *Floating Round to Single-Precision* operation, the target FPR is set to a Quiet NaN
  - FPSCR[FR, FI] ← 0
  - FPSCR[FPRF] ← the class of the result (Quiet NaN)
- If the operation is a convert to 32-bit integer operation, the target FPR is set as follows:
  - $FPR(FRT)_{0:31}$ ← undefined
  - $FPR(FRT)_{32:63}$ are set to the most positive 32-bit integer if the operand in FPR(FRB) is a positive number or +∞, and to the most negative 32-bit integer if the operand in FPR(FRB) is a negative number, -∞, or NaN.
  - FPSCR[FR, FI] ← 0
  - FPSCR[FPRF] ← undefined

- If the operation is a compare:
  - FPSCR[FR, FI, C] ← unchanged
  - FPSCR[FPCC] ← unordered
- If software explicitly requests the exception:
  - FPSCR[FR, FI, FPRF] are as set by the **mtfsf**, **mtfsfi**, or **mtfsb1** instruction.

11.4.2 Zero Divide Exception

A Zero Divide exception occurs when an **fdiv**[**s**] instruction is executed with a zero divisor value and a finite nonzero dividend value. This exception also occurs when a *Reciprocal Estimate* instruction (**fres** or **frsqrte**) is executed with an operand value of zero.

11.4.2.1 Action

The action to be taken depends on the setting of FPSCR[ZE].

When Zero Divide exception is enabled (FPSCR[ZE] = 1) and Zero Divide occurs, the following actions are taken:

- The Zero Divide exception bit is set.

  $FPSCR_{ZX} \leftarrow 1$
- $FPR(FRT)_{0:31} \leftarrow$ unchanged
- FPSCR[FR, FI] ← 0
- FPSCR[FPRF] ← unchanged

When Zero Divide exception is disabled (FPSCR[ZE] = 0) and zero divide occurs, the following actions are taken:

- The Zero Divide exception bit is set.

  $FPSCR_{ZX} \leftarrow 1$
- FPR(FRT) ← ±Infinity (the sign is determined by the XOR of the signs of the operands)
- FPSCR[FR, FI] ← 0
- FPSCR[FPRF] ← class and sign of the result (±Infinity)

11.4.3 Overflow Exception

Overflow occurs when the magnitude of what would have been the rounded result, if the exponent range were unbounded, exceeds that of the largest finite number of the specified result precision.

11.4.3.1 Action

The action to be taken depends on the setting of FPSCR[OE].

When Overflow exceptions re enabled (FPSCR[OE] = 1) and exponent overflow occurs, the following actions are taken:

- Overflow Exception is set

  FPSCR[OX] ← 1
- For double-precision arithmetic instructions, the exponent of the normalized intermediate result is adjusted by subtracting 1536.

- For single-precision arithmetic instructions and the **frsp** instruction, the exponent of the normalized intermediate result is adjusted by subtracting 192.
- FPR(FRT) ← adjusted rounded result
- FPSCR[FPRF] ← class and sign of the result (±Normal Number)

When Overflow Exception is disabled (FPSCR[OE] = 0) and overflow occurs, the following actions are taken:

- Overflow Exception is set

  FPSCR[OX] ← 1
- Inexact Exception is set

  FPSCR[XX] ← 1
- The result is determined by the rounding mode (FPSCR[RN]) and the sign of the intermediate result as follows:
  - Round to Nearest
    Store ± Infinity, where the sign is the sign of the intermediate result
  - Round toward Zero
    Store the format's largest finite number with the sign of the intermediate result
  - Round toward +Infinity
    For negative overflow, store the format's most negative finite number; for positive overflow, store +Infinity
  - Round toward –Infinity
    For negative overflow, store –Infinity; for positive overflow, store the largest finite number of the format
- FPR(FRT) ← result
- FPSCR[FR] ← undefined
- FPSCR[FI] ← 1
- FPSCR[FPRF] ← class and sign of the result (±Infinity or ±Normal Number)

11.4.4 Underflow Exception

Underflow Exception is defined separately for the enabled and disabled states:

- Enabled:
  Underflow occurs when the intermediate result is "Tiny."
- Disabled:
  Underflow occurs when the intermediate result is "Tiny" and there is "Loss of Accuracy."

A "Tiny" result is detected before rounding, when a nonzero intermediate result computed as though both the precision and the exponent range were unbounded would be less in magnitude than the smallest normalized number.

If the intermediate result is "Tiny" and Underflow Exception is disabled (FPSCR[UE] = 0), the intermediate result is denormalized (See *Normalization and Denormalization* on page 149) and rounded (See "Rounding" on page 150) before being placed into the target FPR.

"Loss of Accuracy" is detected when the delivered result value differs from what would have been computed were both the precision and the exponent range unbounded.

11.4.4.1 Action

The action to be taken depends on the setting of FPSCR[UE].

When Underflow exception is enabled (FPSCR[UE] = 1) and exponent underflow occurs, the following actions are taken:

- Underflow Exception is set
  FPSCR[UX] ← 1
- For double-precision arithmetic instructions, the exponent of the normalized intermediate result is adjusted by adding 1536
- For single-precision arithmetic instructions and the **frsp** instruction, the exponent of the normalized intermediate result is adjusted by adding 192
- The adjusted rounded result is placed into the target FPR

  FPSCR[FPRF] ← class and sign of the result (±Normalized Number)

  Programming Note: The FR and FI bits are provided to allow the Enabled exception type Program interrupt, when taken because of an Underflow Exception, to simulate a 'trap disabled' environment. That is, the FR and FI bits allow the Enabled exception type Program interrupt to unround the result, thus allowing the result to be denormalized.

When Underflow Exception is disabled (FPSCR[UE] = 0) and underflow occurs, the following actions are taken:

- Underflow Exception is set

  FPSCR[UX] ← 1
- FPR(FRT) ← rounded result
- FPSCR[FPRF] ← class and sign of the result (±Normalized Number, ±Denormalized Number, or ±Zero)

11.4.5 Inexact Exception

An Inexact Exception occurs when either of the following conditions occur during rounding:

- The rounded result differs from the intermediate result, assuming both the precision and the exponent range of the intermediate result to be unbounded. In this case, the result is said to be inexact. If the rounding causes an enabled Overflow Exception or an enabled Underflow Exception, an Inexact Exception also occurs only if the significands of the rounded result and the intermediate result differ.)
- The rounded result overflows and Overflow Exception is disabled.

11.4.5.1 Action

The action to be taken does not depend on the setting of FPSCR[XX].

When Inexact Exception occurs, the following actions are taken:

- Inexact Exception is set

  FPSCR[XX] ← 1
- FPR(FRT) ← rounded or overflowed result
- FPSCR[FPRF] ← class and sign of the result

  Programming Note: In some implementations, enabling Inexact Exceptions may degrade performance more than does enabling other types of floating-point exception.

11.5 Exception Priorities for Floating-Point Load and Store Instructions

The following prioritized list of exceptions may occur as a result of the attempted execution of any *Floating-Point Load* and *Store* instruction.

1. Debug (Instruction Address Compare)
2. Instruction TLB Error (all types)
3. Instruction Storage Interrupt (all types)
4. Program (Illegal Instruction)
5. Floating-Point Unavailable
6. Program (Unimplemented Operation)
7. Data TLB Error (all types)
8. Data Storage (all types)
9. Alignment
10. Debug (Data Address Compare, Data Value Compare)
11. Debug (Instruction Complete)

If an instruction causes both a Debug (Instruction Address Compare) exception, and a Debug (Data Address Compare) or Debug (Data Value Compare) exception, and does not cause any exception listed in items 2–9, both exceptions can be generated and recorded in the Debug Status Register (DBSR). A single Debug interrupt results.

11.6 Exception Priorities for other Floating-Point Instructions

The following prioritized list of exceptions may occur as a result of the attempted execution of any floating-point instruction other than a load or store.

1. Debug (Instruction Address Compare)
2. Instruction TLB Error (all types)
3. Instruction Storage Interrupt (all types)
4. Program (Illegal Instruction)
5. Floating-Point Unavailable
6. Program (Unimplemented Operation)
7. Program (Enabled)
8. Debug (Instruction Complete)

11.7 QNaN

If any of the source operands is a NaN, either signaling (SNaN) or quiet (QNaN), the result will be that NaN with the high-order fraction bit forced to 1 (that is, forced to a QNaN). The precedence, in decreasing order, is FRA, FRB, FRC. The resultant QNaN is only truncated on an **frsp**[**.**] instruction, in which case the most significant 35 bits are copied to the target, with the least significant 29 forced to zero.

Table 11-4. QNaN Result

R_a	R_b	R_c	Resultant QNaN[a]
NaN	X	X	R_a
—	—	X	R_b[b]
—	—	NaN	R_c

a.High-order fraction bit is forced to a 1
b.**frsp**: Result is $(FRB)_{0:11}$ || 1 || $(FRB)_{13:34}$|| $^{29}0$?

11.8 Updating FPRs on Exceptions

The target FPR is never updated on enabled invalid exceptions and enabled divide by zero exceptions. This requirement exists because an instruction may potentially use one of the source registers as a target register, yet it is necessary that the trap handler be able to examine and act upon the source operands.

In all other cases, a floating-point exception does not block the writing of the target FPR.

11.9 Floating-Point Status and Control Register

The computational instructions modify the FPSCR. With the exception of instructions which write directly to the FPSCR, none of the noncomputational instructions modify the FPSCR.

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. $FPSCR_{0:23}$ are status bits. $FPSCR_{24:31}$ are control bits.

The exception bits in the FPSCR (bits 3:12, 21:23) are sticky; that is, once set to 1 they remain set to 1 until they are set to 0 by an **mcrfs**, **mtfsfi**, **mtfsf**, or **mtfsb0** instruction. The exception summary bits FPSCR[FX, FEX, VX] are not considered as "exception bits," and only FPSCR[FX] is sticky.

FPSCR[FEX, VX] are simply ORs of other FPSCR bits. Therefore, these bits are not listed among the FPSCR bits affected by the various instructions.

FPSCR[FPRF], which contains five result flag bits, is set for arithmetic, rounding, and conversion instructions based on the class of the result value placed into the target FPR. If any portion of a result is undefined, the value placed into FPSCR[FPRF] is undefined. *Table 11-5* describes how the values of the result flags in FPSCR[FPRF] correspond to the result value classes.

Table 11-5. FPSCR[FPRF] Result Flags

Result Flags					Result Value Class
C	<	>	=	?	
1	0	0	0	1	Quiet NaN
0	1	0	0	0	–Infinity

Table 11-5. FPSCR[FPRF] Result Flags

Result Flags					Result Value Class
C	<	>	=	?	
0	1	0	0	0	–Normalized Number
1	1	0	1	0	–Denormalized Number
1	0	0	1	0	–Zero
0	0	0	0	0	+Zero
1	0	1	0	0	+Denormalized Number
0	0	1	0	0	+Normalized Number
0	0	1	0	1	+Infinity

Figure 4-2 on page 143 illustrates the FPSCR.

11.10 Updating the CR

Architecturally, excepting floating-point instructions do not block the updating of the CR in the PPC440 processor core. However, the PPC440 FPU blocks CR updates and requires software assistance to make them.

11.10.1 CR Fields

The CR fields are modified by various floating-point instructions.

Figure 11-1. Condition Register (CR)			
0:3	CR0	Condition Register Field 0	
4:7	CR1	Condition Register Field 1	
8:11	CR2	Condition Register Field 2	
12:15	CR3	Condition Register Field 3	
16:19	CR4	Condition Register Field 4	
20:23	CR5	Condition Register Field 5	
24:27	CR6	Condition Register Field 6	
28:31	CR7	Condition Register Field 7	

11.10.2 Updating CR Fields

The floating-point compare instructions **fcmpo** and **fcmpu** specify a CR field that is updated with the compare results. update a field (specified by the instruction) of the CR.

Table 11-6 illustrates the bit encodings for a CR field containing the results of an **fcmpo** and **fcmpu** instruction.

Table 11-6. Bit Encodings for a CR Field

CR Field (Bit)	Description
0	*Floating-Point Less Than* (FL) Floating-point compare: (FRA) < (FRB)
1	*Floating-Point Greater Than* (FG) Floating-point compare: (FRA) > (FRB)
2	*Floating-Point Equal* (FE) Floating-point compare: (FRA) = (FRB)
3	*Floating-Point Unordered* (FU) Floating-point compare: One or both of (FRA) or (FRB) is a NaN.

The **mcrfs** instruction moves a specified FPSCR field into a CR field.

11.10.3 Generation of QNaN Results

If a disabled Invalid Operation exception is caused by operating on a NaN, the value returned follows the rules indicated in *Table 11-4* on page 259.

If the exception was not caused by operating on a NaN, a QNaN must be generated. The generated QNaN has a sign bit of 0, an exponent of all 1s, a high-order fraction bit of 1 with all other fraction bits of 0: 0x7FF8000000000000.

12. Timer Facilities

The PPC440EPx/GRx provides four timer facilities: a time base, a Decrementer (DEC), a Fixed Interval Timer (FIT), and a Watchdog Timer. These facilities, which share the same source clock frequency, can support:

- Time-of-day functions
- General software timing functions
- Peripherals requiring periodic service
- General system maintenance
- System error recovery

These timer facilities are part of the PPC440 processor core. Refer to the *PPC440 Processor User's Manual* for details.

13. General Purpose Timers

The General Purpose Timer (GPT) is a system timer with seven maskable compare registers and a 32-bit time base counter. Each compare register has a corresponding GPT interrupt to the UIC. GPT interrupts can be generated for a specific count by a match between the contents of a compare register and the time base counter. GPT interrupts may also be generated on a specific interval by masking individual bits of a compare register. The following sections provide an overview, programming steps and register descriptions.

13.1 GPT Features

- 32-bit time base counter driven by the OPB bus clock
- OPB slave interface for access to all control, timer and status registers which provide direct control of all GPT functions
- Seven compare timers and corresponding interrupt outputs
- Programmable time base register (sets the time base counter)
- Maskable time-base comparison support for each compare timer
- Programmable compare timer values
- Enable/disable control of all compare interrupts
- Mask control of interrupt status bits

13.2 Time Base Counter

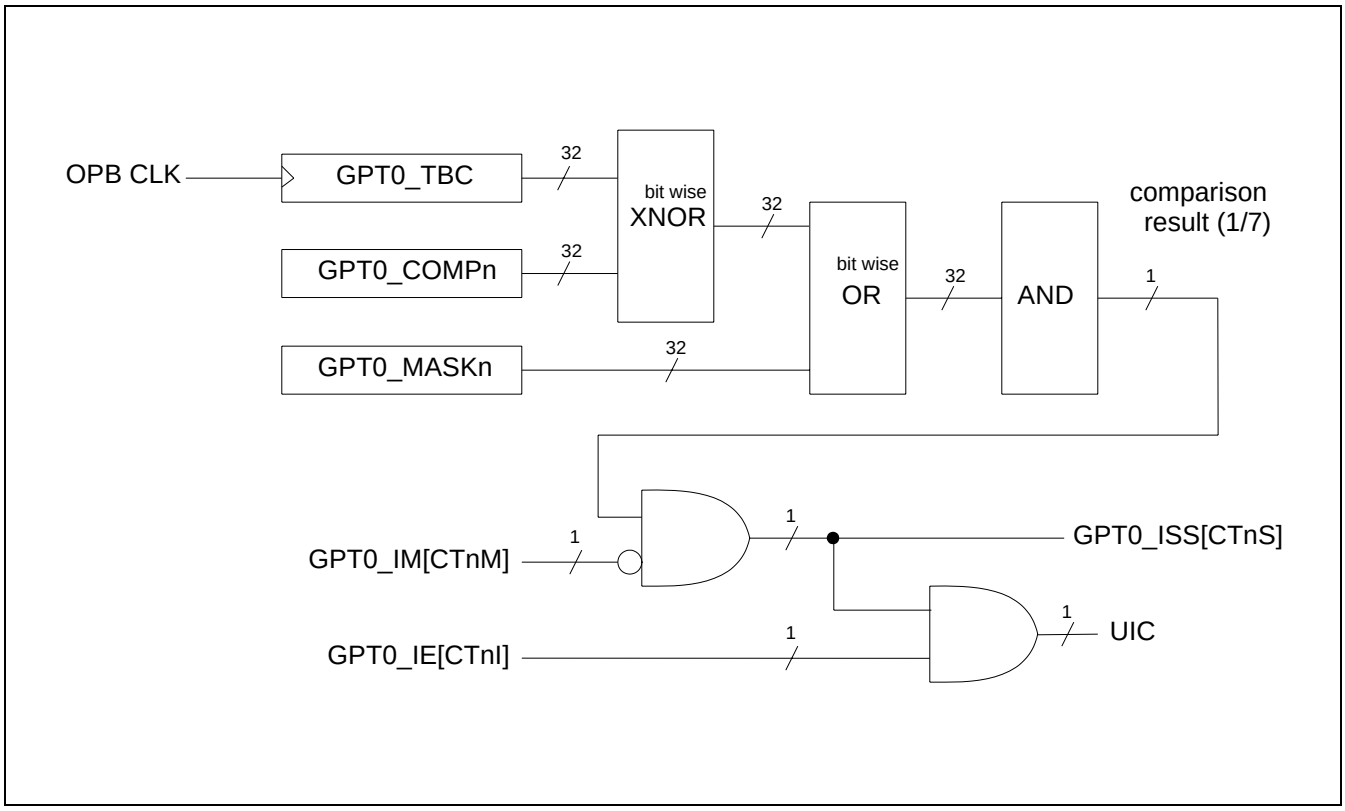
The Time Base Counter (TBC) is both an OPB register and an unsigned counter, and provides the reference time for all compare timers. It increments by one with each clock period and is 32-bit wide. When the time base counter is at its maximum value (all bits set to 1) it rolls back to zero upon the next clock.

The TBC is synchronously reset to zero upon a full chip reset. It may be read and written via software through the OPB interface. When written the new value is stored with the next rising edge of OPBClk.

13.3 Compare Timers

The time base counter, GPT0_TBC, is incremented each OPB clock period and evaluated for equivalence to the compare registers as illustrated in *Figure 13-1*. The XNOR identifies bits in the time base counter that are equivalent to corresponding bits in a compare register GTP0_COMP0:6. The 32-bit output of the XNOR is OR'ed with a 32-bit compare mask GPT0_MASK0:6. Bits set to 1 in the compare mask are masked and are not evaluated for equivalence. The comparison result of the 32-bit input AND reduces the masked result of the OR to a single bit indicating equivalence if set to 1. The status register GPT0_ISS records the comparison if the corresponding compare result is unmasked by the interrupt mask register field GPT0_IM[CTnM] = 0. To generate a GPT interrupt the interrupt enable bit GPT0_IE[CTnI] = 1 must also be enabled. *Figure 13-1* illustrates the comparison of the time base counter to a compare register.

Figure 13-1. Time Base Counter and Compare Register



13.3.1 Compare Timer Interrupt

The following are steps for enabling a GPT interrupt:

1. Set the corresponding compare register (GPT0_COMPn) to the desired compare value.
2. Set the corresponding compare mask register (GPT0_MASKn) with the desired mask bit pattern.
3. Unmask the GPT interrupt by clearing the interrupt mask bit (GPT0_IM[CTnM]=0).
4. Enable the GPT interrupt by setting the enable bit (GPT0_IE[CTnI]=1).
5. Configure the UIC to enable a GPT interrupt (see *Universal Interrupt Controller* on page 205).

Note: Seven separate interrupt lines, (UIC 8, 9, 16, 17, 21, 22, 23) one for each of the seven compare timers are implemented.

13.4 GPT Registers

The GPT device control registers listed in *Table 13-1* are accessed using the move from device control register (**mfdcr**) and move to device control register (**mtdcr**) instructions. All GPT registers are memory mapped and accessed via load/store instructions at the address of the register. The registers are accessed from the OPB on 32-

bit boundaries relative to the configurable base address. The GPT Interrupt Status Register (GPT0_IS) bits are either set or cleared when written, depending upon which one of two addresses are used. All other registers are both read and write accessible in the normal manner.

Table 13-1. GPT Registers

Mnemonic	Register	Address	Access	Page
GPT0_TBC	Time Base Counter	0x1 EF60 0000	R/W	267
GPT0_IM	GPT Interrupt Mask	0x1 EF60 0018	R/W	268
GPT0_ISS	GPT Interrupt Status (Set bits if write 1)	0x1 EF60 001C	R/W	268
GPT0_ISC	GPT Interrupt Status (Clear bits if write 1)	0x1 EF60 0020	R/W	268
GPT0_IE	GPT Interrupt Enable	0x1 EF60 0024	R/W	269
GPT0_COMP0	Compare Timer 0	0x1 EF60 0080	R/W	270
GPT0_COMP1	Compare Timer 1	0x1 EF60 0084	R/W	270
GPT0_COMP2	Compare Timer 2	0x1 EF60 0088	R/W	270
GPT0_COMP3	Compare Timer 3	0x1 EF60 008C	R/W	270
GPT0_COMP4	Compare Timer 4	0x1 EF60 0090	R/W	270
GPT0_COMP5	Compare Timer 5	0x1 EF60 0094	R/W	270
GPT0_COMP6	Compare Timer 6	0x1 EF60 0098	R/W	270
GPT0_MASK0	Compare Mask (Compare Timer 0)	0x1 EF60 00C0	R/W	270
GPT0_MASK1	Compare Mask (Compare Timer 1)	0x1 EF60 00C4	R/W	270
GPT0_MASK2	Compare Mask (Compare Timer 2)	0x1 EF60 00C8	R/W	270
GPT0_MASK3	Compare Mask (Compare Timer 3)	0x1 EF60 00CC	R/W	270
GPT0_MASK4	Compare Mask (Compare Timer 4)	0x1 EF60 00D0	R/W	270
GPT0_MASK5	Compare Mask (Compare Timer 5)	0x1 EF60 00D4	R/W	270
GPT0_MASK6	Compare Mask (Compare Timer 6)	0x1 EF60 00D8	R/W	270
GPT0_DCT0	Down Count Timer	0x1 EF60 0110	R/W	270
GPT0_DCIS	Down Count Timer Interrupt Status	0x1 EF60 011C	R/W	271

13.4.1 GPT Time Base Counter Register (GPT0_TBC)

GPT time base counter register (GPT0_TBC) is used by the compare timers as a reference for determining event occurrences and for software to use as a general timer.

Figure 13-2. GPT Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base

13.4.2 GPT Interrupt Mask Register (GPT0_IM)

GPT interrupt mask register (GPT0_IM) bits correspond to the compare timer interrupt masks. The register bits mask both the setting of the corresponding GPT0_IS bits and the interrupt output signals to the UIC. If masked, GPT0_IS bits are not set, and interrupt signals are not generated (even if the GPT0_IE bits are enabled). For interrupt signals to be active, the GPT0_IM bits must be reset (not masked) and the GPT0_IE bits must be enabled.

Figure 13-3. GPT Interrupt Mask Register (GPT0_IM)

0:15		Reserved	
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled	
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled	
18	CT2M	Compare Timer 2 Interrupt Mask 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled	
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled	
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled	
21	CT5M	Compare Timer 5 Interrupt Mask 0 Compare timer 5 interrupt mask disabled 1 Compare timer 5 interrupt mask enabled	
22	CT6M	Compare Timer 6 Interrupt Mask 0 Compare timer 6 interrupt mask disabled 1 Compare timer 6 interrupt mask enabled	
23:31		Reserved	

13.4.3 GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

GPT interrupt status register (GPT0_ISS/ICC) bits correspond to the compare timer interrupt status. The GPT Interrupt status bits for the compare timers are set when a valid comparison is made, and the compare interrupt is enabled.

GPT0_ISS can be accessed through address offset 0x1C, which provides a normal read access and a "Write-Set" access, allowing individual status bits to be set through a write access. Any status bits written to 1 are set (forced to 1), while bits written to 0 remain unchanged (0 or 1).

Offset 0x20 (GPT0_ISC) provides a normal read access and a "Write-Clear" access, which allows individual status bits to be reset through a write access. Any status bits written to 1 are cleared (forced to 0), while bits written to 0 remain unchanged (0 or 1).

Figure 13-4. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)			
0:15		Reserved	
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled	
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled	
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled	
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled	
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled	
21	CT5S	Compare Timer 5 Interrupt Status 0 Compare timer 5 interrupt status disabled 1 Compare timer 5 interrupt status enabled	
22	CT6S	Compare Timer 6 Interrupt Status 0 Compare timer 6 interrupt status disabled 1 Compare timer 6 interrupt status enabled	
23:31		Reserved	

13.4.4 GPT Interrupt Enable Register (GPT0_IE)

GPT interrupt enable register (GPT0_IE) bits correspond to the compare timer interrupt enable bits. When set, GPT0_IE bits prevent the corresponding compare interrupts from activating the GPT UIC interrupts, even if the interrupt mask (GPT0_IM) bits are set; however, these bits have no effect on the corresponding interrupt status (GPT0_IS) bits.

Figure 13-5. GPT Interrupt Enable Register (GPT0_IE)			
0:15		Reserved	
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled	
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled	
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled	
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled	

20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled	
21	CT5I	Compare Timer 5 Interrupt Enable 0 Compare timer 5 interrupt enable disabled 1 Compare timer 5 interrupt enable enabled	
22	CT6I	Compare Timer 6 Interrupt Enable 0 Compare timer 6 interrupt enable disabled 1 Compare timer 6 interrupt enable enabled	
23:31		Reserved	

13.4.5 GPT Compare Timer Registers (GPT0_COMP0:GPT0_COMP6)

Each GPT compare timer register (GPT0_COMP0:GPT0_COMP6) is programmed with the value that is continually compared to the TBC value, as filtered through each MASK register. The width of each GPT0_COMP0:GPT0_COMP6 is 32 bits.

Figure 13-6. GPT Compare Timer Register (GPT0_COMP0 - GPT0_COMP6)

0:31	COMP	Compare Timer	

13.4.6 GPT Compare Mask Registers (GPT0_MASK0:GPT0_MASK6)

GPT compare mask registers (GPT0_MASK0:GPT0_MASK6) bits are used by the compare timers to mask off the comparison (i.e., force a valid compare) of individual bits when the comparison function is performed. For bits that are set, a valid compare is always assumed, regardless of the actual value of these bits in the GPT0_COMP0:GPT0_COMP6 or GPT0_TBC registers. The width of each implemented Mask Register is 32 bits.

Figure 13-7. GPT Compare Mask Register (GPT0_MASK0:GPT0_MASK6)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.

13.4.7 GPT Down Count Timer (GPT0_DCT0)

The GPT0_DCT0 is loaded with the time or count value to be loaded into the down counter. It is a full 32-bit quantity that is decremented each OPBClk cycle until the counter reaches zero, where it stops until another value is loaded. If the register is loaded with 0xFFFFFFFF, the counter will not count.

Figure 13-8. Down Count Timer Register (GPT0_DCT0)

0:31	DCT	Time value

13.4.8 GPT Down Count Timer Interrupt Status (GPT0_DCIS)

The GPT0_DCIS is used to determine that the down count timer caused the interrupt.

Figure 13-9. Down Count Timer Register (GPT0_DCIS)

0	DCIS	Down Count Interrupt Status 0 Counter not expired 1 Counter expired	Writing a 1 resets the bit.
1:31		Reserved	

14. Clocking

Clocking in the PPC440EPx/GRx is highly configurable and supports a wide range of clock ratios on the internal and external buses. PPC440EPx/GRx clocking and power-on reset (CPR) registers allow for flexible power-on configuration using the IIC bootstrap controller, as well as for later adjustment after the PPC440EPx/GRx has begun operation. See *Section 8.2 Bootstrap Options* on page 189 for bootstrap options and instructions on how to change clock setting after a system reset. Maximum performance at different operating frequencies is possible through the support of non integral relative frequencies for CPU and PLB clocks, including but not limited to ratios such as 3:1, 4:1, 5:1, 5:2, and 7:2.

Two PLLs are used in the PPC440EPx/GRx, which support both System and Ethernet clocking:

1. System PLL - source for CPU and PLB clocks, OPB, serial and external bus clocks
2. Ethernet PLL (fixed value PLL) source for Ethernet clock

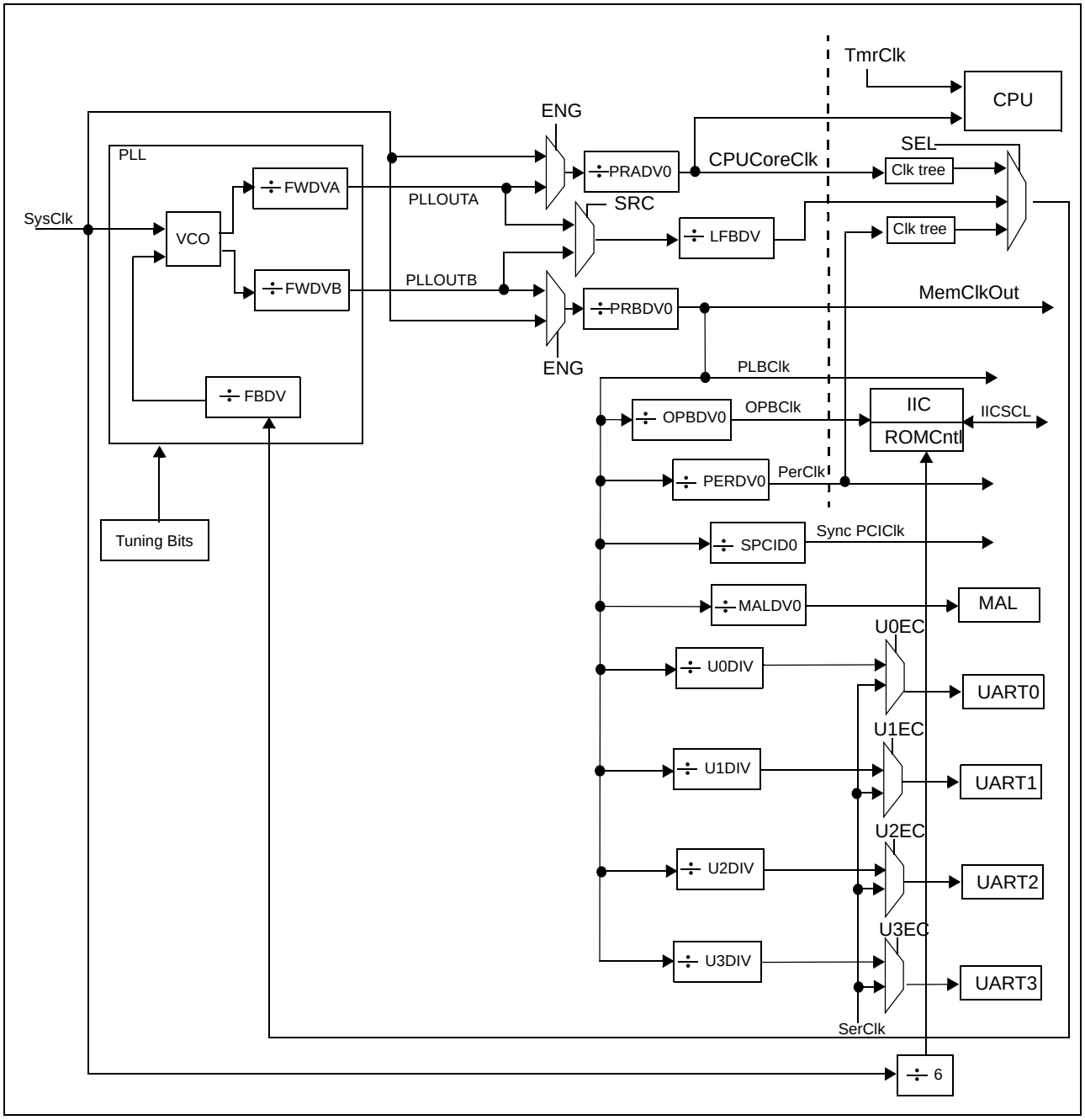
The control and configuration necessary for the system PLL is described in the following sections. Be sure to properly filter both the analog Vdd and GND inputs used by each PLL as described in detail in the PPC440EPx/GRx data sheet.

PLL operation is controlled by a set of registers defined in *Clocking Registers* on page 284. The Voltage Controlled Oscillator (VCO) contained within each PLL is required to operate within the range of 600-1334 MHz. *System Clock Ratio Examples* on page 279 describe the VCO, CPU, PLB, OPB and EBC frequencies set by the clocking registers. *Figure 14-1* shows the PPC440EPx/GRx clocking structure for the system PLL.

14.1 System Clocking

Figure 14-1 illustrates the clocks related to general system operation of the PPC440EPx/GRx. Note that the CPU frequency is generated from PLLOUTA, while the PLB frequency is generated from PLLOUTB.

Figure 14-1. PPC440EPx/GRx System Clocking



The system clock provided to the PPC440EPx/GRx must be a minimum of 33MHz in order for the on-chip PLL to achieve a stable lock. An input clock above 67MHz is not supported, and the provided clock must be monotonic, and have acceptable jitter and slew rate as specified in the PPC440EPx/GRx data sheet.

14.1.1 Feedback Selection

The PPC440EPx/GRx clocking logic operates in four different modes: PLL bypass mode, PLL engaged with PLL local feedback (PLLOUTA or PLLOUTB), PLL engaged with CPU clock feedback, and PLL engaged with peripheral clock feedback. These modes are shown in *Table 14-1*.

Table 14-1. PPC440EPx/GRx Clocking Modes

CPR0_PLLC0[ENG]	CPR0_PLLC0[SEL]	Mode
0	xxx	Bypass
1	000	PLL Local feedback
1	001	CPU clock feedback
1	101	PER clock feedback

The PLL operates in Bypass mode upon entering Reset (see *Reset and Initialization* on page 169 for additional information on reset). In this mode the system reference clock, SYSClk, replaces the output of the PLL. The frequencies of the various clocks are SYSclk divided by all the divisors in that clock's path. The PLL stays in Bypass mode until the initial configuration is determined from either the default serial rom straps or the values read in from the serial ROM device. When the initial configuration has been determined, set up, and the clocks are stable, reset is released to the rest of PPC440EPx/GRx.

The PPC440EPx/GRx system clocking logic is designed to use one of three feedback paths for the SYS PLL, selected by CPR0_PLLC0 [SEL]. Most users will find that local feedback provides the greatest flexibility in choosing CPU and PLB clock ratios. Users who require that the external bus clock, PerClk, be both frequency- and phase-aligned with SysClk will need to use PerClk for feedback. Phase alignment between SysClk and the PerClk clock permits the use of the external master interface with external masters that cannot use PerClk for synchronization. These external masters will normally source the external bus clock themselves and provide it to the PPC440EPx/GRx SysClk input. CPU clock feedback is provided as a compatibility mode with the PPC440GP.

For PerClk and SysClk to be phase aligned, the clock divisors must be selected such that the frequencies are the same. Doing so requires that the CPR0_PLLD[FBDV] bits be set to a divide-by-one, and that PLB and other clocks derived from PLLOUTB are a multiple of PerClk. In the absence of any system requirement for phase alignment between PerClk and SysClk, users are encouraged to use PLL local feedback mode.

Note: In Bypass mode the DDR SDRAM interface may not work, because the DDR delay line is only designed to work between 133 and 166 MHz and allowed system reference clocks are 33 to 67MHz. See *DDR SDRAM Controller* on page 405 for additional information.

PPC440EPx clocking mode can be changed by programming the clocking control registers to the desired values for the new clocking mode and then programming the CPU to issue a Chip Reset.

14.1.2 VCO Frequency and 'M' Value for SYS PLL

For any acceptable input SysClk frequency, the SYS PLL VCO frequency is set by the total product of divisor circuits used in the path from VCO output back to VCO input, multiplied by the system clock frequency. The product of the divisor circuits, both inside the PLL and in the external divisor circuits such as those that generate the PLB, OPB, and PerClk, is referred to as the M multiplier value. This value must be known in order to properly set the TUNE bits.

The M multiplier for the SYS PLL can be calculated using any of the equations shown in *Table 14-2*, depending upon the choice made for feedback as shown in the tables that follow.

Table 14-2. Clock Frequencies and M multiplier

Feedback Choice	M multiplier equation
PLL Local	M = FBDV x LFBDV x FWDVA (CPR0_PLLC0[SRC]=0)
PLL Local	M = FBDV x LFBDV x FWDVB (CPR0_PLLC0[SRC]=1)
CPU clock	M = FBDV x FWDVA x PRADV0
PerClk Clock	M = PERDV0 x FWDVB x PRBDV0 (FBDV=1)

As an example of the effect that various settings have on clock frequencies and the M multiplier, consider the System PLL configured first with PLL Local feedback as shown in *Table 14-3*, then with CPU feedback as shown in *Table 14-4*, and finally with PerClk feedback as shown in *Table 14-5*..

Table 14-3. System PLL Configuration Using PLL Local Feedback (CPR0_PLLC0[SEL] = 000)

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2 (PRADV0 =1)	CPU = 400MHz
FWDVB = 6 (PRBDV0 =1)	PLB = 133.3MHz
OPBDV0 = don't care	OPB = don't care
PERDV0 = don't care	PerClk = don't care
LFBDV = 3 (CPR0_PLLC0[SRC]=0)	
FBDV = 4	M = 24
M = 24	TUNE = 1100111100

Table 14-4. System PLL Configuration Using CPU Feedback (CPR0_PLLC0[SEL] = 001)

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2 (PRADV0 =1)	CPU = 400MHz
FWDVB = 6 (PRBDV0 =1)	PLB = 133.3MHz
OPBDV0 = don't care	OPB = don't care
PERDV0 = don't care	PerClk = don't care
FBDV = 12	M = 24
M = 24	TUNE = 1100111100

Table 14-5. System PLL Configuration Using PerClk Feedback (CPR0_PLLC0[SEL] = 101)

PLL Settings	Resulting Configuration
SYSCLK = 33.3MHz	VCO = 800MHz
FWDVA = 2 (PRADV0 =1)	CPU = 400MHz
FWDVB = 6 (PRBDV0 =1)	PLB = 133.3MHz
OPBDV0= don't care	OPB = don't care
PERDV0 = 4	PerClk = 33.3 MHz
FBDV = 1	M = 24
M = 24	TUNE = 1100111100

14.1.3 SYS PLL TUNE Setting

The SYS PLL must be provided with different TUNE bit settings based upon the VCO frequency and 'M' value that result from the system configuration. *Table 14-6* describes how to determine the proper CPR0_PLLC0[TUNE] bit settings for appropriate application.

Table 14-6. CPR0_PLLC0[TUNE] Bit Settings

'M' Value Range	VCO Frequency Range	TUNE									
		0	1	2	3	4	5	6	7	8	9
$9 \leq M \leq 10$	$600MHz < VCO \leq 900MHz$	0	1	0	0	1	1	0	1	1	0
$10 < M \leq 22$	$600MHz < VCO \leq 900MHz$	1	0	0	0	1	1	1	0	0	0
	$900MHz < VCO \leq 1066MHz$	1	0	1	0	1	1	1	0	0	0
$22 < M \leq 40$	$600MHz < VCO \leq 900MHz$	1	1	0	0	1	1	1	1	0	0
	$900MHz < VCO \leq 1334MHz$	1	1	1	0	1	1	1	1	0	0

14.1.4 IIC Bootstrap Controller Clocking

SysClk is divided by 6 to provide a clock for the IIC bootstrap controller. This clock is internal to the PPC440EPx/GRx and cannot be adjusted. See *Bootstrap Controller* on page 187 for additional information. Also, the IIC bootstrap controller further divides its clock by 128. Thus, for SysClk frequencies at or below 66.67MHz, a 100KHz serial ROM can be used.

14.1.5 Clocks For Off Chip Use

The differential MemClk signal for the DDR SDRAM chips is connected directly to the PLB Clock. The PerClk is generated by the system clocking logic for use by an external bus master or other synchronous device. The UART serial clocks can be generated either from an internal clock divisor circuit or from the SerClk chip input. These clocks are presented internally to the UART cores, which can be further subdivided.

14.1.6 SYS PLL Strapping

System clocking is primarily controlled by the CPR0_PLLC0, CPR0_PLLD, CPR0_OPBD, CPR0_PERD, CPR0_SPCID, and SDR0_CP440 registers. These registers are normally initialized at power on time using the IIC bootstrap controller and an external serial ROM. The SDR0_SDSTP0 and SDR0_SDSTP1 registers are also initialized during this process. If a serial ROM is not present at power on time, then the pre-programmed defaults will be used.

System reset always re-loads the SDR0_SDSTP0, SDR0_SDSTP1, SDR0_CUST0 and SDR0_CUST1 registers using the values programmed into the serial bootstrap ROM.

Several different bit values in the clocking registers are required to correctly configure the chip in a way that will allow reliable operation of SYS PLL and clock divisor circuitry. When setting these values be careful to avoid accidentally configuring the PPC440EPx/GRx in an unusable state. See the following section for assistance in selecting acceptable system clocking divisor ratios.

14.1.7 PLL Bypass (Emulation Mode)

The CPR0_PLLD[ENG] bit is provided to disable the SYS PLL if it is intentionally being used outside of its operating range. The most common reason for using this mode is in emulation systems used in product development, where the SYS CLK input to PPC440EPx/GRx is set to a frequency well below 33MHz, perhaps as low as 1MHz. Forcing the SYS PLL into bypass mode directly feeds SYSClk to the divisor logic, allowing parts of the PPC440EPx/GRx to operate well outside of its supported range (see the PPC440EPx/GRx data sheet for supported range).

14.1.8 CPU / PLB Frequency N:1 Setting

The PPC440EPx/GRx clocking logic allows for CPU:PLB clock frequencies that are in the ratio of N:X, where N = {1,2,...,etc} and X = {1, 2}. The CPU clock is considered to be an integral multiple of PLB when X = 1, as in the example where CPU = 500 and PLB = 166 (N:X = 3:1). A non-integral CPU to PLB relationship can be seen when X = 2, as in the example where CPU = 415 and PLB =166 (N:X = 5:2).

The CPU:PLB ratio is based on the FWDVA, FWDVB, PRADV0 and PRBDV0 divisors: CPU:PLB=FWDVB*PRBDV0:FWDVA*PRADV0. Only N:1 and N:2 ratios that do not exceed the minimum or maximum CPU, PLB and DDR clock frequencies are supported. If the calculated CPU:PLB ratio is not N:1 or N:2 but can be reduced to N:1 or N:2, the ratio is supported as long as the resulting N is a whole number. See *System Clock Ratio Examples* on page 279 to determine valid CPU and PLB frequencies and their resultant CPU:PLB, N:X, ratio.*Table 14-7* indicates the results of programming SDR0_CP440[Nto1] to either 0 or 1, depending upon whether the CPU:PLB ratio is N:1.

Table 14-7. SDR0_CP440[Nto1] Settings

CPU:PLB Ratio	SDR0_CP440[Nto1] = 0	SDR0_CP440[Nto1] = 1
CPU:PLB Ratio is N:1 (X = {1})	Slight performance loss as unnecessary clock resynchronization occurs at CPU:PLB interface	Optimal performance for N to 1 mode
CPU:PLB Ratio is not N:1 (X = {2})	Optimal performance for non N to 1 mode	System failure can occur

14.1.9 Choosing System Clock Ratios

Table 14-8 describes the rules that apply when choosing acceptable system clock ratios and frequencies for the PPC440EPx/GRx. Please see the current data sheet as different PPC440EPx/GRx revisions may have different limits than those described here.

Table 14-8. System Clock Rules

Rule	Value
CPU frequency must be less than or equal to	Max CPU frequency - see PPC440EPx/GRx data sheet
PLB frequency must be less than or equal to	Max PLB frequency - see PPC440EPx/GRx data sheet
CPU:PLB ratio is N:X with {N, X} being integer and X less than or equal to	2
VCO frequency must be greater than or equal to	600MHz
VCO frequency must be less than or equal to	Max supported VCO - see PPC440EPx/GRx data sheet

Table 14-9 describes the equations that can be used to determine the resulting VCO, CPU, PLB frequencies based on selected divisor settings.

Table 14-9. Equations to Determine VCO, CPU, PLB Frequency

Feedback Selection	Equations
PLL Local	M = FBDV x LFBDV x FWDVA (CPR0_PLLC0[SRC]=0) or M = FBDV x LFBDV x FWDVB (CPR0_PLLC0[SRC]=1) VCO = SysClk x M CPU = VCO / FWDVA / PRADV0 PLB = VCO / FWDVB / PRBDV0
CPU clock	M = FBDV x FWDVA x PRADV0 VCO = SysClk x M CPU = VCO / FWDVA / PRADV0 PLB = VCO / FWDVB / PRBDV0
PerClk	M = PERDV0 x FWDVB x PRBDV0 VCO = SysClk x M CPU = VCO / FWDVA / PRADV0 PLB = VCO / FWDVB / PRBDV0

14.1.10 System Clock Ratio Examples

Example tables are provided in the following sections for the case where SysClk = 33.3MHz and PLL Local feedback, CPU feedback or PerClk feedback are used. These tables describe possible combinations of clocking ratios which can be achieved using the bit values in CPR0_PLLC0, CPR0_PLLD, CPR0_PRIMAD, CPR0_PRIMBD, CPR0_PERD,CPR0_OPBD and CPR0_SPCID.

Note: The value of PLLOUTA and PLLOUTB must not exceed the rated speed of the processor core.

14.1.10.1 PLL Local Feedback Example

As shown in *Table 14-10*, when using PLL local feedback, M and VCO can be determined once specific values for the FWDVA, LFBDV and FBDV divisors are chosen. Therefore, the PLL local feedback table is organized so that each row represents a specific combination of FWDVA, LFBDV and FBDV divisor. From these specified values, M and VCO are known and listed. Given a value of 1 for PRAVD0 the CPU frequency is also known and listed.

For each combination of these divisors, there are up to seven different columns on the right hand of the table which indicate the possible PLB frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVB and PRBDV0 divisor values.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 600 MHz or above 1334 MHz, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Table 14-10 provides clock ratio listing with PLL Local feedback, PLLOUTA (CPR0_PLLC0[SRC]=0), for CPR0_PRIMAD0[PRADV0] = 1, SysClk = 33.3 MHz, 600 MHz ≤ VCO ≤ 1334 MHz, CPU ≤ 667 MHz, 133 MHZ ≤ PLB ≤ 166 MHZ, with CPU to PLB clock ratios of 5:2, 3:1, 7:2 and 4:1. See the data sheet for maximum CPU and VCO frequencies.

Table 14-10. Clock Ratio Listing With PLL Local Feedback, SysClk = 33.3 MHz

FWDVB ====>						2	3	4	5	6	7	8
PRBDV0	FWDVA	FBDV x LFBDV	M	VCO	CPU	PLB Clock Frequency (MHz)						
1	2	2-8	1-16	>600	-							
1	2	9	18	600.0	300.0			150.0				
1	2	10	20	666.7	333.3			166.6	133.3			
1	2	11	22	733.3	366.7				146.4			
1	2	12	24	800.0	400.0				160.0	133.3		
1	2	13	26	866.7	433.3					144.4		
1	2	14	28	933.3	466.7					155.3	133.3	
1	2	15	30	1000.0	500.0					166.6	142.9	
1	2	16	32	1066.7	533.3						152.4	
1	1	20	20	666.7	666.7			166.6	133.3			
2	2	20	40	1333.3	666.7			166.6	133.3			

14.1.10.2 CPU Feedback Example

As shown in *Table 14-11*, when using CPU feedback, M, VCO and CPU can be determined once specific values for the FWDVA, PRADV0 and FBDV divisors are chosen. Therefore, the CPU feedback table is organized so that each row represents a specific combination of FWDVA, PRADV0 and FBDV divisor. From these specified values, M, VCO and CPU are known and listed.

For each combination of these divisors, there are up to seven different columns on the right hand of the table which indicate the possible PLB frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVB and PRBDV0 divisor values.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 600 MHz or above 1334 MHz, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Table 14-11 provides clock ratio listing with CPU feedback for CPR0_PRIMAD0[PRADV0]=1, SysClk = 33.3MHz, 600MHz ≤ VCO ≤ 1334MHz, CPU ≤ 667MHz, 133 MHZ ≤ PLB ≤ 166 MHZ, with CPU to PLB clock ratios of 5:2, 3:1, 7:2 and 4:1. See the data sheet for maximum CPU and VCO frequencies.

Table 14-11. Clock Ratio Listing With CPU Feedback, SysClk = 33.3MHz

FWDVB ====>						2	3	4	5	6	7	8
PRBDV0	FWDVA	FBDV	M	VCO	CPU	PLB Clock Frequency (MHz)						
1	2	1-8	1-16	>600	-							
1	2	10	20	666.7	333.3			166.6	133.3			
1	2	12	24	800.0	400.0				160.0	133.3		
1	2	13	26	866.7	433.3					144.4		
1	2	14	28	933.3	466.7						133.3	
1	2	15	30	1000.0	500.0					166.6	142.9	
1	2	16	32	1066.7	533.3						152.4	133.3
1	1	20	20	666.7	666.7			166.6	133.3			
2	2	20	40	1333.3	666.7			166.6	133.3			

14.1.10.3 PerClk Feedback Example

As shown in the System Clock Rules table, when using PerClk feedback, M, VCO and PLB can be determined once specific values for the FWDVB, PRBDV0, PERDV0 divisors are chosen. Therefore, the PerClk feedback table is organized so that each row represents a specific combination of FWDVB and PERDV0 with PRBDV0 =1. From these specified values, M, VCO and CPU are known and listed.

Since PERDV0 can vary from a divide by 1 up to a divide by 4, the possible values are 1, 2, 3, 4,. Furthermore, when using external bus feedback, PerClk has the same frequency as SysClk. This in turn implies that FBDV is 1 and PLB = SysClk x PERDV0. *Table 14-12* demonstrates this fact by listing possible PLB frequencies which are all multiples of SysClk (that is, 133.3MHz, assuming SysClk = 33.3MHz.

For each combination of these divisors, there are up to eight different columns on the right hand of the table which indicate the possible CPU frequencies that result from the VCO frequency determined for that row, depending upon a specific FWDVA divisor value.

Entries that result from obviously unusable configurations are left empty, including those that result in VCO frequencies below 600MHz or above 1334MHZ, and excessive CPU and/or PLB frequencies. In several cases the table will indicate identical CPU and PLB frequencies, however in each case there will be a different VCO frequency that results.

Note: *Table 14-12* provides clock ratio listing with PerClk feedback for CPR0_PRIMAD0[PRADV0]=1, SysClk = 33.3MHz, 600MHz ≤ VCO ≤ 1334 MHz, CPU ≤ 667MHz, 133MHZ ≤ PLB ≤ 166MHZ, with CPU to PLB clock ratios of 5:2, 3:1, 7:2 and 4:1. See the data sheet for maximum CPU and VCO frequencies.

Table 14-12. Clock Ratio Listing With PerClk feedback, SysClk = 33.3MHz

FWDVA ====>						1	2	3	4	5	6	7	8
PRBDV0	PERDV0	FWDVB	M	VCO	PLB	CPU Clock Frequency (MHz)							
1	1	2-8		>600	-								
1	2	2-8		>600	-								
1	3	2-5		>600	-								
1	3	6	18	600.0									
1	3	7	21	700.0									
1	3	8	24	800.0									
1	4	2-4		>600									
1	4	5	20	666.7	133.3	666.7	333.3						
1	4	6	24	800.0	133.3		400.0						
1	4	7	28	933.3	133.3		466.7						
1	4	8	32	1066.7	133.3		533.3						
2	4	5	40	1333.3	133.3		666.7		333.3				

Table 14-13 shows the CPU:PLB ratios that result from the indicated FWDVA and FWDVB divisors when using PerClk feedback when SysClk is 33MHz. Several ratios are left blank because they are illegal.

Table 14-13. CPU:PLB Ratio

	FWDVA Divisor							
FWDVB	1	2	3	4	5	6	7	8
2								
3	3:1							
4	4:1							
5		5:2						
6								
7		7:2						
8								

14.2 PCI Clocking

The following clocks are related to the PCI logic:

- The on-chip PLB clock
- The on-chip synchronous PCI clock
- An external asynchronous PCI clock, PCIClk

The PPC440EPx/GRx supports only asynchronous mode. In asynchronous PCI mode, the asynchronous PCI clock attaches to logic that interfaces with the PCI bus. The synchronous clock must always be provided to the PCI logic, even when the asynchronous PCI clock is used. *Table 14-14* describes the relationships between the synchronous PCI clock, the asynchronous PCI clock, and the PLB clock.

In asynchronous PCI mode, the synchronous PCI clock must meet certain requirements. The following equation describes the relationship that must be maintained between the asynchronous PCI clock and synchronous PCI clock. Select an appropriate PCI:PLB ratio to maintain the relationship:

$$\text{AsyncPCIClk} - 1\text{MHz} \leq \text{SyncPCIclock} \leq (2 \times \text{AsyncPCIClk}) - 1\text{MHz}$$

Table 14-14 lists supported and commonly used combinations of synchronous and asynchronous PCI clocks. In general, higher synchronous PCI clock frequencies provides better performance, while lower synchronous PCI clock frequencies minimize power consumption.

Table 14-14. Example Synchronous PCI Clock Frequencies in Asynchronous Mode

Asynchronous PCI Frequency (External PCI clock)	Synchronous PCI Frequency (Internal PCI clock)	PLB Frequency	Sync PCI:PLB Ratio (CPR0_SPCID[SPCID0])
20 MHz	33.3 MHz	133.3 MHz	1:4
33.3 MHz	44.4 MHz	133.3 MHz	1:3
	33.3 MHz	133.3 MHz	1:4
	55.3 MHz	166.6 MHz	1:3
	41.65 MHz	166.6 MHz	1:4
40 MHz	44.4 MHz	133.3 MHz	1:3
	50.8 MHz	152.4 MHz	1:3
	55.3 MHz	166.6 MHz	1:3
66.6 MHz	66.6 MHz	133.3 MHz	1:2
	76.2 MHz	152.4 MHz	1:2
	83.3 MHz	166.6 MHz	1:2

14.2.1 PCI Adapter Applications

Because various systems run PCI expansion buses at different PCI frequencies, several PCI clock frequencies may need to be supported when the PPC440EPx/GRx is used in PCI adapters.

Asynchronous PCI mode uses an externally provided PCI clock that does not interact with an on-chip PLL, so there is no lower frequency limit imposed by loss of PLL lock. However, the requirements resulting from the relationship between the synchronous and asynchronous PCI clocks must still be satisfied.

Note: Satisfying the equation in *Section 14.2* on page 282 presents a potential problem. The divisor selection needed to set an acceptable synchronous PCI clock for a 33 MHz asynchronous PCI clock differs from the selection for a 66 MHz asynchronous PCI clock. External logic is required to detect the state of the M66 pin on the PCI adapter interface and select appropriate PPC440EPx/GRx divisor values during system reset.

14.3 Clocking Registers

Table 14-16 summarizes the indirectly accessed device control registers that control clocking and power on reset in PPC440EPx/GRx.These registers are accessed by using the configuration address and data registers described in *Table 14-15*.

Table 14-15. PPC440EPx/GRx Clocking Control Register Access

Mnemonic	Register	Address	Access	Page
CPR0_CFGADDR	Clocking Configuration Address Register	0x000C	R/W	284
CPR0_CFGDATA	Clocking Configuration Data Register	0x000D	R/W	284

Table 14-16. PPC440EPx/GRx Clocking Control Registers

Mnemonic	Register	Offset	Access	Page
CPR0_CLKUPD	Clocking Update Register	0x0020	R/W	285
CPR0_PLLC0	PLL Control Register	0x0040	R/W	285
CPR0_PLLD0	PLL Divisor Register	0x0060	R/W	286
CPR0_PRIMAD0	Primary A Divisor Register	0x0080	R/W	287
CPR0_PRIMBD0	Primary B Divisor Register	0x00A0	R/W	288
CPR0_OPBD0	OPB Clock Divisor Register	0x00C0	R/W	288
CPR0_PERD0	Peripheral Clock Divisor Register	0x00E0	R/W	288
CPR0_MALD	MAL Clock Divisor Register	0x0100	R/W	289
CPR0_SPCID	Sync PCI Clock Divisor Register	0x0120	R/W	289
CPR0_ICFG	Clock/Power Configuration Register	0x0140	R/W	176

14.3.1 Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR)

The clock/power-on reset configuration address register (CPR0_CFGADDR) is a 32-bit register used to access the PPC440EPx/GRx CPR0 configuration registers.

Figure 14-2. Clock/Power-On Reset Configuration Address Register (CPR0_CFGADDR)

0:16		Reserved	
17:31		Offset	

14.3.2 Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA)

The clock/power-on reset configuration data register (CPR0_CFGDATA) is a 32-bit register used to access the PPC440EPx/GRx CPR0 configuration registers.

Figure 14-3. Clock/Power-On Reset Configuration Data Register (CPR0_CFGDATA)

0:31			

14.3.3 Clocking Update Register (CPR0_CLKUPD)

The clocking update register (CPR0_CLKUPD) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register.

Note: This register is documented here for reference purposes only and should not be used to change clocking modes. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-4. Clocking Update Register (CPR0_CLKUPD)

0	BSY	Clocking Subsystem Busy 0 Clocking subsystem is stable 1 Clocking subsystem is making changes.	In Read Access Mode. When CPR0_CLKUPD[BSY=1], software needs to wait until CPR0_CLKUPD[BSY]=0 before initiating any additional changes.
	CUD	Clocking Update Delay 0 Clocking update delay disabled 1 Clocking update delay enabled	In Write Access Mode. When CPR0_CLKUPD[CUD] is set to 1, then: Clocking update action is taken. Instruct the clocking logic to begin changing clocks to the newly programmed divider values. Result. System clocks will be momentarily stopped and then restarted using the currently programmed values in the CPR0 divider registers
1	CUI	Clocking Update Immediate 0 Clocking update immediate disabled 1 Clocking update immediate enabled	When you read it this bit is always 0. In Write Access Mode. When CPR0_CLKUPD[CUI]=1: Clocking update action taken. Allow the currently programmed values in all CPR0 registers to take effect immediately. Result. The clocking subsystem is now programmed according to the current settings of all CPR0 registers
2:31		Reserved	

Note: The CPR0_CLKUPD register is unique, in that writes will automatically result in an update to the clocking subsystem, while reads of the same bit(s) will report a status. Please note that this register is documented here for reference purposes only and should not be used to change clocking modes.

14.3.4 PLL Control Register (CPR0_PLLC0)

The PLL control register (CPR0_PLLC0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-5. PLL Control Register (CPR0_PLLC0)

0	RST	Reset 0 PLL allowed to lock 1 PLL is forced into reset.	For the CPR clocking logic, the reset value of the RST bit is the complement of the ENG bit
1	ENG	Engage 0 SysClk is the source for primary forward divisor. 1 PLL's VCO is the source for primary forward divisor.	
2	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB	

3:4		Reserved	
5:7	SEL	Feedback Selection 000 PLL output (A or B) 001 CPU 101 PERClk	All other combinations are reserved Selection of PLL output A or B implies that the feedback clock is taken exactly at or close to the PLL output, and not at the end of a repowered clock tree. Using this feedback source implies that the PLL is not being used to adjust the generated clocks to be phase aligned with SysClk. The specific clock used for feedback is controlled by CPR0_PLLCR[SRC] bit 2
8:21		Reserved	
22:31	TUNE	TUNE bits	

14.3.5 PLL Divisor Register (CPR0_PLLD0)

The PLL divisor register (CPR0_PLLD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-6. PLL Divisor Register (CPR0_PLLD)

0:2		Reserved	
3:7	FBDV	PLL Feedback Divisor 00000 PLL Feedback Divisor = 32 00001 PLL Feedback Divisor = 1 00010 PLL Feedback Divisor = 2 00011 PLL Feedback Divisor = 3 .... 11111 PLL Feedback Divisor = 31	
8:11		Reserved	
12:15	FWDVA	PLL Forward Divisor A 0000 PLL Forward Divisor A = 16 0001 PLL Forward Divisor A = 1 0010 PLL Forward Divisor A = 2 0011 PLL Forward Divisor A = 3 0100 PLL Forward Divisor A = 4 0101 PLL Forward Divisor A = 5 0110 PLL Forward Divisor A = 6 0111 PLL Forward Divisor A = 7 1000 PLL Forward Divisor A = 8 1001 Reserved 1010 PLL Forward Divisor A = 10 1011 Reserved 1100 PLL Forward Divisor A = 12 1101 Reserved 1110 PLL Forward Divisor A = 14 1111 Reserved	
16:20		Reserved	

21:23	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 Reserved 010 PLL Forward Divisor B = 2 011 PLL Forward Divisor B = 3 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 5 110 PLL Forward Divisor B = 6 111 PLL Forward Divisor B = 7	
24:25		Reserved	
26:31	LFBDV	PLL Local Feedback Divisor 00_0000 PLL local feedback divisor = 64 00_0001 PLL local feedback divisor = 1 00_0010 PLL local feedback divisor = 2 .... 11_1111 PLL local feedback divisor = 63	The LFBDV is outside of PLL and is used to allow the PLL to lock using feedback directly from a PLL output. Program this divider to a value equivalent to the divide from the PLL to some clock to allow the PLL to be switched to use feedback from that clock. This allows the PLL to compensate for the latency associated with that clock's tree.

14.3.6 Primary A Divisor Register (CPR0_PRIMAD0)

The primary A divisor register (CPR0_PRIAMD) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA register. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-7. Primary A Divisor Register (CPR0_PRIMAD0)

0:4		Reserved	
5:7	PRADV0	PLL Primary Divisor A 000 PLL Primary Divisor A = 8 001 PLL Primary Divisor A = 1 010 PLL Primary Divisor A = 2 011 PLL Primary Divisor A = 3 100 PLL Primary Divisor A = 4 101 PLL Primary Divisor A = 5 110 PLL Primary Divisor A = 6 111 PLL Primary Divisor A = 7	**Note:** If CPR0_ICFG[RLI] = 0, then the reset value for PLL Primary Divisor A is 1. PRADV0 = 1 is the recommended setting. This setting ensures, PLLOUTA clock does not exceed the rated speed of the processor core (CPU Clk). The output of CPR0_PLLC[FWDVA] which is PLLOUTA is the same frequency as the CPU CLK when PRADV0 is set to 1.
8:31		Reserved	

14.3.7 Primary B Divisor Register (CPR0_PRIMBD0)

The primary B divisor register (CPR0_PRIMBD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-8. Primary B Divisor Register (CPR0_PRIMBD0)

0:4		Reserved	
5:7	PRBDV0	PLL Primary Divisor B 000 PLL Primary Divisor B = 8 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 011 PLL Primary Divisor B = 3 100 PLL Primary Divisor B = 4 101 PLL Primary Divisor B = 5 110 PLL Primary Divisor B = 6 111 PLL Primary Divisor B = 7	PRBDV0 = 1 is the recommended setting. This setting ensures, PLLOUTB clock does not exceed the rated speed of the processor core (CPU Clk). The output of CPR0_PLLC[FWDVB] which is PLLOUTB is the same frequency as the PLB CLK when PRBDV0 is set to 1.
8:31		Reserved	

14.3.8 OPB Clock Divisor Register (CPR0_OPBD0)

The OPB clock divisor register (CPR0_OPBD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-9. OPB Clock Divisor Register (CPR0_OPBD0)

0:5		Reserved	
6:7	OPBDV0	OPB Clock Divisor 0 00 OPB clock divisor 0 = 4 01 OPB clock divisor 0 = 1 10 OPB clock divisor 0 = 2 11 OPB clock divisor 0 = 3	
8:31		Reserved	

14.3.9 Peripheral Clock Divisor Register (CPR0_PERD0)

The peripheral clock divisor register (CPR0_PERD0) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-10. Peripheral Clock Divisor Register (CPR0_PERD0)			
0:4		Reserved	
5:7	PERDV0	Peripheral Clock Divisor 0 000 Reserved 001 Peripheral clock divisor 0 = 1 010 Peripheral clock divisor 0 = 2 011 Peripheral clock divisor 0 = 3 100 Peripheral clock divisor 0 = 4 101 Peripheral clock divisor 0 = 5 110 Reserved 111 Reserved	
8:31		Reserved	

14.3.10 MAL Clock Divisor Register (CPR0_MALD)

The MAL clock divisor register (CPR0_MALD) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers. See *Section 8.2* on page 189 for instructions on changing clocking mode.

Figure 14-11. MAL Clock Divisor Register (CPR0_MALD)			
0:5		Reserved	
6:7	MALDV0	MAL Clock Divisor 0 00 MAL clock divisor 0 = 4 01 MAL clock divisor 0 = 1 10 MAL clock divisor 0 = 2 11 MAL clock divisor 0 = 3	Set MALDV0 = CPR0_OPBD0[OPBDV0]. Typically MALDV0=OPBDV0=2.
8:31		Reserved	

14.3.11 Sync PCI Clock Divisor Register (CPR0_SPCID)

The PCI clock divisor register (CPR0_SPCID) is a 32-bit read/write register which is accessed by using CPR0_CFGADDR and CPR0_CFGDATA registers.

Figure 14-12. Sync PCI Clock Divisor Register (CPR0_SPCID)			
0:5		Reserved	
6:7	SPCID0	PCI Clock Divisor 0 00 PCI clock divisor 0 = 4 01 PCI clock divisor 0 = 1 10 PCI clock divisor 0 = 2 11 PCI clock divisor 0 = 3	
8:31		Reserved	

15. Clock and Power Management

The PPC440EPx/GRx provides a clock and power management (CPM) controller that reduces power dissipation by stopping clocks in unused or dormant functional units. Use of the CPM controller requires careful programming and special consideration to avoid compromising system and functional unit integrity.

15.1 Overview

The CPM controller supports three different types of sleep interfaces to the functional units:

- In a CPM class 1 interface, the CPM_Sleep_N signal is asserted by the CPM controller when a register bit is set by software. The functional unit is unconditionally put to sleep. There is no other communication with the functional unit.
- In a CPM class 2 interface, the functional unit uses a combination of its internal state and external inputs to determine whether or not it can be put to sleep. If sleeping is permitted, the functional unit asserts the Sleep_Req signal to the CPM controller that responds by asserting CPM_Sleep_N if the enable for that unit is set. The CPM_Sleep_N signal to a class 2 unit is deasserted when the CPM controller enable bit for that unit is reset, or when the unit deasserts its Sleep_Req signal.
- The CPM class 3 interface has a CPM_SleepInit signal that is asserted by the CPM controller to request that a functional unit go to sleep. If the unit can sleep, it asserts the Sleep_Req signal to the CPM controller. The CPM_Sleep_N signal is then asserted by the CPM controller to shut off the class 3 clocks in the functional unit. The functional unit or the CPM controller can end the sleep state. If the CPM controller enable bit for the unit is reset, the CPM controller immediately deasserts CPM_SleepInit and CPM_Sleep_N.

15.2 CPM Registers

Table 15-1 lists the registers used to program the CPM controller.

Table 15-1. CPM Registers

Register	Description	DCR Address	Access	(Bits 0:31)
CPM0_ER	CPM0 Enable Register	0x00B0	Read/Write	0x000000000
CPM0_FR	CPM0 Force Register	0x00B1	Read/Write	0x000000000
CPM0_SR	CPM0 Status Register	0x00B2	Read Only	0xFFFFFFFF
CPM1_ER	CPM1 Enable Register	0x00F0	Read/Write	0x000000000
CPM1_FR	CPM1 Force Register	0x00F1	Read/Write	0x000000000
CPM1_SR	CPM1 Status Register	0x00F2	Read Only	0xFFFFFFFF

15.2.1 CPM Enable Registers (CPM0_ER and CPM1_ER)

The CPM0_ER bits enable the process of putting a functional unit to sleep. The class of a unit determines how its interface signals are controlled when the bit associated with the unit is set to 1.

Class 1 — When an associated CPMx_ER bit is set to 1, the CPM_Sleep_N signal to the class 1 unit is asserted. When the bit is set to 0, CPM_Sleep_N is deasserted.

Class 2 — When an associated CPMx_ER bit is set to 1, and the Sleep_Req signal from the class 2 unit is asserted (the unit is requesting sleep state), CPM_Sleep_N to the class 2 unit is asserted. When the bit is set to 0, the CPM_Sleep_N signal is deasserted.

Class 3 — When an associated CPMx_ER bit is set to 1, the CPM_SleepInit signal to the class 3 unit is asserted (the CPM controller is requesting permission to put the unit to sleep). When the class 3 unit activating the Sleep_Req in response, (the unit is giving permission to be put to sleep), CPM_Sleep_N signal to the class 3 unit is asserted. When the bit is set to 0, CPM_SleepInit and CPM_Sleep_N are deasserted.

Figure 15-1. CPM0 Enable Register (CPM0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 1
3	P42OPB0	PLB4 to OPB Bridge 0 (USB 2.0 Device, *PPC440EPx only)*	Class 2
4	UDMA	DMA2PLB4 DMA	Class 2
5	FPU	PPC440 Floating Point Unit *(PPC440EPx only)*	Class 1
6	CPU	440 Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	P32OPB	PLB3 to OPB Bridge	Class 2
9		Reserved	
10	EBC	External Bus Controller	Class 2
11	RGMII	Reduced Gigabit MII Bridge	Class 1
12		Reserved	
13	PLB4	PLB4 Arbiter	Class 2
14	PLB4x3	PLB4 to PLB3 Bridge Controller	Class 2
15	PLB3x4	PLB3 to PLB4 Bridge Controller	Class 2
16	PLB3	PLB3 Arbiter	Class 2
17	NDFC	NAND Flash Controller	Class 2
18		Reserved	
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1

27	UART2	Universal Asynchronous Receiver/Transmitter 2	Class 1
28	UART3	Universal Asynchronous Receiver/Transmitter 3	Class 1
29	EMC1	Ethernet 1	Class 1
30	P42OPB1	PLB4 to OPB 1 Bridge (USB 2.0 Host, *PPC440EPx only)*	Class 2
31	OPB2P4	OPB to PLB4 Bridge (USB 2.0 Host, *PPC440EPx only)*	Class 2

Figure 15-2. CPM1 Enable Register (CPM1_ER)

0	UIC2	Universal Interrupt Controller 2	Class 1
1	SRAM0	Internal SRAM Controller 0	Class 2
2	MAL0	Memory Access Layer	Class 1
3	USB2D0	USB 2.0 Device *(PPC440EPx only)*	Class 1
4	USB2H0	USB 2.0 Host (*PPC440EPx only)*	Class 1
5	CRYP0	Security Engine	Class 1
6	KASU0	Kasumi Engine	Class 1
7:31		Reserved	

15.2.2 CPM Force Register (CPM0_FR and CPM1_FR)

Setting a CPMx_FR bit forces assertion of the CPM_Sleep_N signal to the functional unit. For a class 1 unit, this is equivalent to setting the CPMx_ER bit associated with the unit. For class 2 or class 3 units, CPM_Sleep_N is asserted regardless of the state of the Sleep_Req signal coming from the unit.

Figure 15-3. CPM0 Force Register (CPM0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 1
3	P42OPB0	PLB4 to OPB Bridge 0 (USB 2.0 Device, *PPC440EPx only*)	Class 2
4	UDMA	DMA2PLB4 DMA	Class 2
5	FPU	PPC440 Floating Point Unit (*PPC440EPx only)*	Class 1
6	CPU	440 Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	P32OPB	PLB3 to OPB Bridge	Class 2
9		Reserved	
10	EBC	External Bus Controller	Class 2
11	RGMII	Reduced Gigabit MII Bridge	Class 1
12		Reserved	
13	PLB4	PLB4 Arbiter	Class 2
14	PLB4x3	PLB4 to PLB3 Bridge Controller	Class 2
15	PLB3x4	PLB3 to PLB4 Bridge Controller	Class 2

16	PLB3	PLB3 Arbiter	Class 2
17	NDFC	NAND Flash Controller	Class 2
18		Reserved	
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1
27	UART2	Universal Asynchronous Receiver/Transmitter 2	Class 1
28	UART3	Universal Asynchronous Receiver/Transmitter 3	Class 1
29	EMC1	Ethernet 1	Class 1
30	P42OPB1	PLB4 to OPB Bridge 1 (USB 2.0 Host, *PPC440EPx only)*	Class 2
31	OPB2P4	OPB to PLB4 Bridge (USB 2.0 Host, *PPC440EPx only*)	Class 2

Figure 15-4. CPM1 Force Register (CPM1_FR)

0	UIC2	Universal Interrupt Controller 2	Class 1
1	SRAM0	Internal SRAM Controller 0	Class 2
2	MAL0	Memory Access Layer	Class 1
3	USB2D0	USB 2.0 Device (*PPC440EPx only)*	Class 1
4	USB2H0	USB 2.0 Host, (*PPC440EPx only)*	Class 1
5	CRYP0	Security Engine	Class 1
6	KASU0	Kasumi Engine	Class 1
7:31		Reserved	

15.2.3 CPM Status Register (CPM0_SR and CPM1_SR)

The read-only CPMx_SR shows the current state of all CPM_Sleep_N signals.

The following figures describes CPMx_SR bit assignment and CPM class for each PPC440EPx/GRx functional unit.

Figure 15-5. CPM Status Register (CPM0_SR)			
0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCI	Peripheral Component Interconnect	Class 1
3	P42OPB0	PLB4 to OPB Bridge 0 (USB 2.0 Device, *PPC440EPx only)*	Class 2
4	UDMA	DMA2PLB4 DMA	Class 2
5	FPU	PPC440 Floating Point Unit (*PPC440EPx only)*	Class 1
6	CPU	440 Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	P32OPB	PLB3 to OPB Bridge	Class 2
9		Reserved	
10	EBC	External Bus Controller	Class 2
11	RGMII	Reduced Gigabit MII Bridge	Class 1
12		Reserved	
13	PLB4	PLB4 Arbiter	Class 2
14	PLB4x3	PLB4 to PLB3 Bridge Controller	Class 2
15	PLB3x4	PLB3 to PLB4 Bridge Controller	Class 2
16	PLB3	PLB3 Arbiter	Class 2
17	NDFC	NAND Flash Controller	Class 2
18		Reserved	
19	UIC1	Universal Interrupt Controller 1	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0	Universal Interrupt Controller 0	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1
27	UART2	Universal Asynchronous Receiver/Transmitter 2	Class 1
28	UART3	Universal Asynchronous Receiver/Transmitter 3	Class 1
29	EMC1	Ethernet 1	Class 1
30	P42OPB1	PLB4 to OPB Bridge 1 (USB 2.0 Host, *PPC440EPx only)*	Class 2
31	OPB2P4	OPB to PLB4 Bridge (USB 2.0 Host, *PPC440EPx only)*	Class 2

Figure 15-6. CPM1 Status Register (CPM1_SR)			
0	UIC2	Universal Interrupt Controller 2	Class 1
1	SRAM0	Internal SRAM Controller 0	Class 2
2	MAL0	Memory Access Layer	Class 1
3	USB2D0	USB 2.0 Device (*PPC440EPx only)*	Class 1
4	USB2H0	USB 2.0 Host (*PPC440EPx only)*	Class 1
5	CRYP0	Security Engine	Class 1
6	KASU0	Kasumi Engine	Class 1
7:31		Reserved	

16. Debug Facilities

The debug facilities of the PPC440EPx/GRx include support for several debug modes for debugging during hardware and software development, as well as debug events that allow developers to control the debug process. Debug registers control these debug modes and debug events. The debug registers may be accessed either through software running on the processor or through the JTAG debug port of the PPC440EPx/GRx Embedded Processor. Access to the debug facilities through the JTAG debug port is typically provided by a debug tool. A trace port, which enables the tracing of code running in real time, is also provided.

The JTAG interface is a part of the PPC440 processor. Refer to the *PPC440 Processor User's Manual* for details.

17. Security Function

The built-in security feature (PPC440EPx-S and PPC440GRx-S) parts only is a cryptographic engine attached to the 128-bit PLB with built-in DMA and interrupt controllers.

Features include:

- Federal Information Processing Standard (FIPS) 140-2 design
- Support for an unlimited number of Security Associations (SA)
- Different SA formats for each supported protocol (IPsec/SSL/TLS/sRTP)
- Internet Protocol Security (IPsec) features
  - Full packet transforms, Encapsulated Security Payload (ESP) and Authentication Header (AH)
  - Complete header and trailer processing (IPv4 and IPv6)
  - Multi-mode automatic padding
  - "Mutable bit" handler for AH, including IPv4 option and IPv6 extension headers
- Secure Socket Layer (SSL) and Transport Layer Security (TLS) features
  - Packet transforms
  - One-pass hash-then-encrypt for SSL and TLS packet transforms for inbound packet using Stream Cipher
- Secure Real-Time Protocol (sRTP) features
  - Packet transforms
  - ROC removal and TAG insertion
  - Variable bypass offset of header length per packet
- IPsec/SSL security acceleration function
- DES, 3DES, AES, ARC-4 encryption
  - Does not support hashing zero length messages
- MD-5, SHA-1 hashing, HMAC encrypt-hash and hash-decrypt, and KASUMI
- Public key acceleration for RSA, DSA and Diffie-Hellman
- True or pseudo random number generators
  - Non-deterministic true random numbers
  - Pseudo random numbers with lengths of 8- or 16-bytes
  - ANSI X9.17 Annex C compliant using a DES algorithm
- Interrupt controller
  - Fifteen programmable, maskable interrupts
  - Initiate commands via an input interrupt
  - Sixteen programmable interrupts indicating completion of certain operations
  - All interrupts mapped to one level- or edge-sensitive programmable interrupt output
- DMA controller
  - Autonomous, 4-channel
  - 1024-words (32 bits/word) per DMA transfer
  - Scatter/gather capability with byte aligned addressing

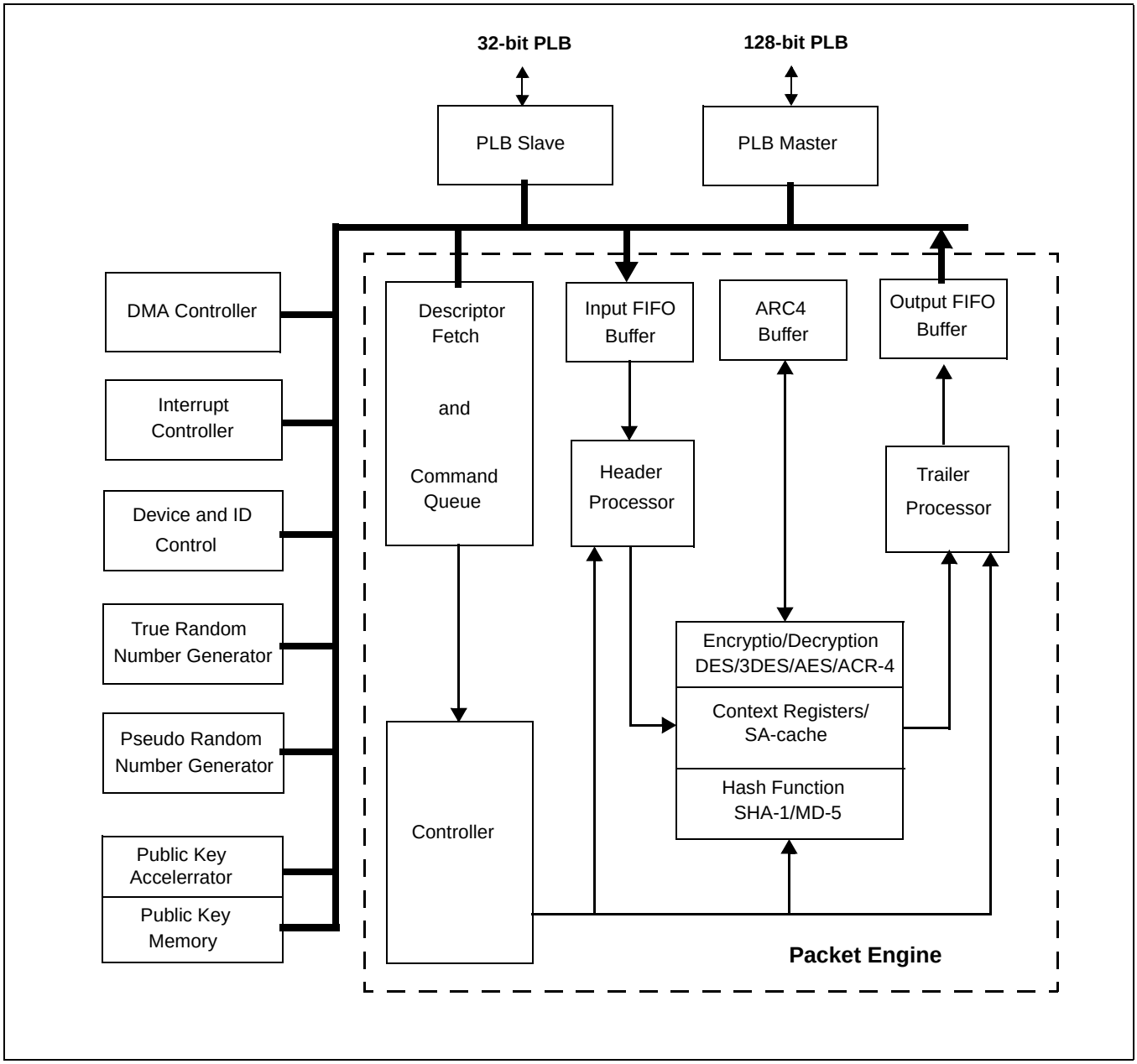
17.1 Functional Description

The following sections provide a brief functional description of the operation of the PPC440EPx/GRx-S security function.

17.1.1 Block Diagram

*Figure 17-1*shows the functional block diagram of the security function within the PPC440EPx/GRx-S.

Figure 17-1. Security Function Block Diagram



17.1.2 PLB Interface

The PLB Slave interface allows the processor to configure the security function and to access functions such as the true random number generator and public key accelerator. The PLB Master interface allows the security function to fetch descriptors and cryptographic context information, and transfer packet data to and from the Packet Engine

17.1.3 Packet Engine

The Packet Engine is the main feature in the security function. It is optimized to offload time-consuming cryptographic operations from the processor and contains functions for both symmetric encryption and hashing. The Packet Engine runs autonomously, performing DMA transfers of control information and payload data to and from the main bus. The DMA process incorporates flow-control to guarantee proper data flow. The Packet Engine allows parallel and pipe lined encryption and hashing operations, significantly reducing the latency and processing time for packets that need multiple operations applied.

17.1.3.1 Input/Output Buffer

The data for the Packet Engine is buffered at both input and output. These buffers decouple the DMA I/O process from the encryption and hash modules inside the Packet Engine. This enables large DMA burst sizes and allows the cryptographic function to keep busy during I/O latency periods. Data is automatically moved from the input buffer through the encryption and hash functions and into the output buffer. If the output buffer is full, the process will pause until data has been read and there is space available in the output buffer.

17.1.3.2 Controller

This is the manager of the Packet Engine that allows it to autonomously process packets from either a Descriptor Ring or a direct command register set. The security function is be configured in one of three command modes.

The first is the Autonomous Ring mode, in which case the security function polls a descriptor ring in memory space. Various polling parameters may be configured at initialization. An input interrupt allows the processor to issue an interrupt to the security function in order to trigger a descriptor fetch.

The second mode is the Target Command mode. A packet command register set within the security function is written by the processor to initiate a packet operation. This eliminates much of the I/O bus overhead of polling for descriptors, but requires the processor to synchronously initiate packet processing.

The third mode is the Direct Host DMA mode. In this mode the processor must provide the packet command register set, the SA and the packet data. This means there are no autonomous read or write transfers; the processor always initiates the transactions.

When a packet operation is complete, the security function provides a result structure that indicates the status and provides the new length, pad result, IPsec next header field, and so on. The security function can write this result to a ring in memory using the PLB Master interface, or the processor can read the result structure from internal security function registers using the PLB Slave interface. An optional interrupt can be generated by the security function at completion of packet processing.

17.1.3.3 Header and Trailer Processor

This header and trailer processors implement all of the protocol header and trailer processing for the security protocols. This includes padding and optional insertion of an IV at the beginning of a packet. When implementing IPsec operations, these functions performs all IPsec header and trailer insertion and removal for both the ESP and AH as shown in the *Table 17-1* and *Table 17-2*:

Table 17-1. IPsec ESP Header Processing

Element	Size	Outbound	Inbound
SPI (Security Parameters Index)	32 bits	Insert.	Extract, verify against SA record.
Replay Counter/Sequence Number	32 bits	Increment and insert.	Extract, verify against expected count and 64-bit window mask; update count and mask after authentication passes.
IV (Initialization Vector)	variable	Insert random value.	Extract and load into cryptographic function.
Padding	0–255B	Insert padding up to 255 bytes.	Strip padding (selectable).
Next header	8 bits	Insert into pad trailer field.	Extract and report in result descriptor.
ICV (Integrity Check Value)	96 bits	Calculate and insert.	Extract and verify. Optionally discard.

Table 17-2. IPsec AH Processing

Element	Size	Outbound	Inbound
Outer IP header	20–60 B	Update length, next header, and header checksum in IP header. Zero the "mutable bit" fields as HMAC is calculated.	Parse the outer IP header and options (IPv4) or extension headers (IPv6) to locate AH header. Zero the "mutable bit" fields as HMAC is calculated.
SPI (Security Parameters Index)	32 bits	Insert.	Extract, verify against SA record.
Replay Counter/Sequence Number	32 bits	Increment and insert.	Extract, verify against expected count and 64-bit window mask.
Next header	8 bits	Insert into AH header field.	Extract and report in result descriptor.
ICV (Integrity Check Value)	96 bits	Calculate and Insert.	Extract and verify. Optionally discard.

17.1.3.4 Encryption/Decryption

Performs high-speed encrypt/decrypt operations, including multiple block cipher and stream cipher algorithms.

Block ciphers: AES, DES and Triple DES. All four standard modes are supported:
- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- 64-bit Output FeedBack (OFB) for DES/Triple-DES
- 1-bit, 8-bit and 64-bit Cipher FeedBack (CFB) for DES/Triple-DES and 128-bit CFB for AES.

Stream ciphers: AES Counter and ARC4. Two AES counter modes and two ARC4 modes are supported:
- AES Counter Mode (CTR) as defined for IPsec ESP.
- AES Integer Counter Mode (ICM) as defined for SRTP.
- ARC4 "Stateless" and "Stateful" mode

All AES modes are supported with key lengths of 128-, 192-, and 256-bits.

The high performance encryption/decryption implementations are highly pipelined and execute full 16-round DES in only four clock cycles and 10-round AES in only 10 clock cycles, minimizing packet latency. Key scheduling is automatic and performed in parallel with the encryption/decryption operation.

Key lengths up to 128 bits are supported.

The high performance ARC4 uses a 256-byte RAM for storage of the S-box data and a second 256-byte RAM that functions as shadow S-box. With the availability of two S-box RAMs, two permutations can be formed at the same time to generate the key stream. Hardware support for padding insertion, verification and removal further accelerates the encryption/decryption process.

17.1.3.5 Hash Function

Tightly coupled with the encryption/decryption function, the hash function provides hardware accelerated one-way hashing operations. The security function supports both the MD-5 and SHA-1 algorithms. An HMAC processing unit is included as well, which automatically handles the "outer" hash calculations.

The hash function contains an intelligent “mutable bit” handler in hardware, which adds a significant performance boost to IPsec AH processing. The security function automatically clears the necessary fields in an IPv4 or IPv6 header as well as the appropriate IPv4 Options fields or IPv6 Extension Header fields.

- The Message Authentication Code (MAC) is used for SSL Operations.
- The Hashed MAC (HMAC) is used for Basic, IPsec, TLS and SRTP Operations.

17.1.3.6 Command Queue and Context Registers

The control data that specifies the processing for each packet is provided in two elements: The Packet Descriptor and the Security Association (SA) record.

The real-time operation of the Packet Engine is controlled by packet commands that are written into a command queue that drives the Packet Engine. The command queue can be written by the processor, or filled with descriptor data that is automatically fetched from a descriptor ring in memory by the Packet Engine.

The SA-record is a packed structure that contains the remainder of the information needed by the Packet Engine to process a packet. The data is stored in context registers that can be written by the processor, or filled with SA-record data that is automatically fetched from memory by the Packet Engine.

17.1.3.7 Using the Packet Engine

This section will discuss some general principles and methods for using the Packet Engine efficiently.

Selection of Descriptor Modes

The first decision to be made relates to the Packet Engine's modes of operation.

The major choice is whether the processor must supply the packet descriptor, SA-record data and packet data, or whether data is fetched autonomously by the Packet Engine.

The Direct Host DMA mode allows the processor to have full control over the Packet Engine through the PLB Slave Interface. However, this mode can be very inefficient since much communication is required between the processor and the security processor.

For the PLB Master mode, there is a choice among enabling Descriptor Ring processing, Autonomous Ring mode, or whether the application will target-write commands (Target Command mode) directly into the on-chip Command Queue. This Command Queue appears as a set of 5-word packet-command registers.

In general, the Autonomous Ring mode is more efficient because it allows the security processor and the processor to operate asynchronously, and allows queuing of multiple packets to be processed. This minimizes starving of the security processor and provides the highest possible throughput.

If Descriptor Ring mode is enabled, then another mode choice is related to the grouping of the Packet Descriptors, the Packet Data, and the SA structure. Each of these elements can be in distinct locations, with the Descriptor pointing to both Packet Data and SA location. Or, the data elements may be linked together so that one piece follows another.

Another choice if Descriptor Ring mode is enabled, is whether the processor will write Packet Descriptors directly into the on-chip Command Queue (one entry) or whether the security processor should fetch them from a External Packet Descriptor Ring (PDR) in external memory (Descriptor Fetch Engine enabled). This choice is made with the PE Ring Size register (0 specifies that the processor writes directly to the internal Command Queue).

Assuming the External PDR mode is used, a decision must be made as to how the security processor is made aware of new descriptors appearing on the PDR. The two choices are Polling or Interrupt.

In the Polling configuration, the security processor simply polls the PDR until it finds one or more entries that have the ownership bits set to the security processor. The frequency of polling, and therefore the amount of bus bandwidth that is consumed, is configurable in the PE Ring Poll register. Separate controls are provided for normal polling and for poll re-tries when the descriptor read is not ready.

In the Interrupt configuration, the Host populates one or more Packet Descriptors and then issues an interrupt to the security processor to tell it to fetch the descriptor(s) and begin processing. This mode usually imposes less bus overhead on the system. It also offers controlled processing latency, since the processor specifies when descriptors will be processed.

Result Descriptors (Autonomous Ring mode only)

When the security processor finishes processing a packet, it writes a Result Descriptor to a Result Descriptor Ring (RDR). The RDR can be thought of as a mirror to the PDR. The user specifies on what bus and at what base address the RDR ring exists. This flexibility is provided so that the results can be efficiently posted to the processor without requiring the processor to perform Master reads.

Generally, if the processor does not write packet descriptors to the internal Command Queue, then the PDR and RDR should be overlaid on top of each other. This minimizes the memory consumed for descriptors and reduces the memory bus utilization. If the PDR and RDR are in separate external locations, then an additional update is also required to the ownership bits in the PDR to prevent the security processor from re-processing old descriptors. (This additional update may be disabled in the CRYP0_PE_DMA_CF register if the processor wishes to prevent ring wrapping.)

SA-Record Storage

In Autonomous Ring mode and Target Command mode, the SA-records are always stored in a memory area that the security processor can access by means of reads across the PLB Master. Each SA-record is 128-bytes in size, and there is no limit to how many SA's the SECURITY ENGINE can support.

In the Direct Host DMA mode, the security processor cannot perform a master fetch of the SA-record data, so it must be presented by the processor after writing the descriptor command and before writing packet data to the input buffer.

The SA registers are generally accessed by the processor only in Direct Host DMA Mode. Although they contain meaningful values in Autonomous Ring mode and Target Command mode, it is recommended that these registers be used only for debugging purposes in these modes.

17.1.4 DMA Controller

The security function incorporates a high-performance 4-channel 32-bit DMA controller which can be used to efficiently move data between memory, the Packet Engine, and the other internal security function modules. The four channels are: descriptor fetch, data I/O, scatter/gather descriptor fetch, and user-programmable.

The user programmable channel is provided for the user to initiate PLB Master DMA transfers to internal locations.

17.1.5 Public Key Accelerator and Public Key Memory

The Public Key Accelerator (PKA) provides high-performance large-vector arithmetic functions, such as multiply, add, subtract, or exponentiation. Vector sizes up to 2048 bits are supported. The Public Key Memory (PKM) stores the intermediate results during large number arithmetic operations and holds the final result.

17.1.5.1 Operation

The Public Key Co-Processor (PKCP) operates directly on the Public Key Memory (PKM). Operand and result vectors are stored in the PKM. The vectors are sequentially cycled through the processing engines of the PKCP and intermediate products from large or complex operations are temporarily stored in this PKM.

The processor is responsible for configuring the PKA for the intended operation, providing proper operand data, and allocating space for the result vector. The processor can access the PKCP through the PLB Slave interface to program the PKM vector locations, vector lengths and the function to be performed. It can also read the status and result information. The processor can access the PKM through the PLB Slave interface to write operands and read result vectors.

Once the processor has initiated an operation, the PKCP must have unrestricted access to the PKM RAM.

17.1.5.2 Functional Description

The PKA is configured for an operation through a set of registers accessed from the PLB. Using these registers, the processor specifies the function, along with the length and location of the operand and result vectors. Both operand and result vectors are stored in the PKM external to the PKCP. The PKM resides on the internal bus, allowing the processor to write it with the input operands, and read the result vectors.

The PKA engine accesses these vectors through the memory interface. During computation, the vectors are sequentially cycled through the math processors of the PKCP. Intermediate results from large or complex operations are routinely written back to this memory for temporary storage. Memory allocation must be performed with consideration to its secondary role as the working space for the PKA.

When the processor sets CRYP0_PKA_FUNC[RUN], the PKA engine begins an operation. This bit remains set until the PKA engine has completed the operation and then is cleared. The processor can either poll this register or enable the PKA interrupt in the security processor Interrupt Controller to determine when an operation is complete.

17.1.6 True Random Number Generator

The true random number generator provides a truly random, non-deterministic noise source for the purpose of generating keys, Initialization Vectors (IV's), and other random number requirements.

17.1.7 Pseudo Random Number Generator

The ANSI X9.17 Annex C compliant pseudo random number generator provides a pseudo random source for the purpose of generating Initialization Vectors (IV's) for DES/Triple-DES and AES algorithms to the Packet Engine, and other pseudo random number requirements.

It provides up to 16-bytes of data at a time, enabling the generation of random IVs for DES, Triple-DES and AES on a per packet basis without slowing down the system. The DES block within the PRNG uses a 64-bit LFSR as input data.

Unlike true random number generators, which exploit the randomness that occurs in some physical phenomena, pseudo random number generators are devices or algorithms that output statistically independent and unbiased numbers. In general, a PRNG is a deterministic algorithm that for a truly random binary sequence, outputs a binary sequence which appears to be random.

17.1.8 Interrupt Controller

The interrupt controller provides the ability to initiate descriptor fetches by means a security function input interrupt. This avoids the overhead of frequent polling for valid descriptors across the PLB Master interface.

In addition, under programmable configuration control, additional interrupts may be generated due to completion of certain operations such as *packet processing complete*, *user DMA transfer complete*, *public key accelerator operation complete*, and so on.

The sixteen interrupts that can be delivered to the UIC are shown in *Interrupt Unmasked and Masked Status Registers (CRYP0_INT_UNMSK/MSK)* on page 374.

The CRYP0_INT_EN register provides the mask to select which interrupt source is enabled (see *Table 9-1*, interrupt 23). A pair of status registers, CRYP0_INT_UNMSK and CRYP0_INT_MSK, allows the processor to read the status of any interrupt source before and after the mask is applied.

These interrupts are latched in both the Unmasked and Masked status registers. Clearing the interrupts using the CRYP0_INT_MSK registers resets both Masked and Unmasked latch stages.

17.1.9 Device Controller

The Device Controller contains miscellaneous identification and cryptographic control registers. They uniquely identify the security function version and its incorporated features to allow software to tune its control to the target instantiation and its capabilities.

17.2 Register Interface

The following sections describe the security function register interface in detail. All of the security function registers are 32-bit MMIO registers.

17.2.1 Byte Ordering

Security function registers are implemented and described in little endian format. Therefore, bit 31 is the most significant bit (msb) and bit 0 is the least significant bit (lsb). Big endian software must consider this when accessing the registers. For example, to put the value 44332211 in a security function 32-bit register, big endian software must either do a store (**stw**) of the value 11223344 or do a byte-reversed store (**stwbr**) of the value 44332211. In either case, the value 44332211 appears on the appropriate word of the PLB data bus. Refer to the *Storage Addressing* section of the *PPC440 Processor User's Manual* for a more detailed description of this process.

17.2.2 Register Summary

Table 17-3 contains a summary list of all the registers associated with the PPC440EPx security function.

.

Table 17-3. Security Function Registers

Register Mnemonic	Description	Address	Access	Page
Security Function ControlRegister				
SDR0_CRYP0	CRYP0 SDR Control Register	offset 0x4500	R/W	312
Packet Engine (PE) Command Registers				
CRYP0_PE_CTLST	PE Control/Status	0x0 E010 0000	R/W	313
CRYP0_PE_SOURCE	PE Source Address	0x0 E010 0004	R/W	319
CRYP0_PE_DEST	PE Destination Address	0x0 E010 0008	R/W	319
CRYP0_PE_SA	PE SA Address	0x0 E010 000C	R/W	320
CRYP0_PE_LENGTH	PE Length	0x0 E010 0010	R/W	321
CRYP0_SA_CMD_0	See SA Command 0 Word	0x0 E010 0014	Write only	na
CRYP0_SA_CMD_1	See SA Command 1 Word	0x0 E010 0018	Write only	na
Packet Engine Control Registers				
CRYP0_PE_DMA_CF	PE DMA Configuration	0x0 E010 0040	See register	322
CRYP0_PE_DMA_ST	PE DMA Status	0x0 E010 0044	Read only	324
CRYP0_PE_PDR_BA	PE Packet Descriptor Ring Base Address	0x0 E010 0048	R/W	326
CRYP0_PE_RDR_BA	PE Result Descriptor Ring Base Address	0x0 E010 004C	R/W	327
CRYP0_PE_RING_S	PE Ring Size and Offset	0x0 E010 0050	R/W	328
CRYP0_PE_RING_P	PE Ring Poll	0x0 E010 0054	R/W	329
CRYP0_PE_I_RING	PE Internal Ring Status	0x0 E010 0058	Read only	330
CRYP0_PE_E_RING	PE External Ring Status	0x0 E010 005C	Read only	331
CRYP0_PE_IO_THR	PE I/O Threshold	0x0 E010 0060	R/W	331
CRYP0_PE_GATH	PE Gather Particle Ring Base Address	0x0 E010 0064	R/W	332
CRYP0_PE_SCAT	PE Scatter Particle Ring Base Address	0x0 E010 0068	R/W	332
CRYP0_PE_PT_S	PE Particle Ring Size	0x0 E010 006C	R/W	333
CRYP0_PE_PT_CFG	PE Particle Ring Configuration	0x0 E010 0070	R/W	333
Packet Engine Security Association (SA) and DMA Registers				
CRYP0_PE_PR_SCA	PE Particle Descriptor Source Address	0x0 E010 0500	R/W (**Note**)	334
CRYP0_PE_PR_SCC	PE Particle Descriptor Source Control	0x0 E010 0504	R/W (**Note**)	335
CRYP0_PE_PR_DTA	PE Particle Descriptor Destination Address	0x0 E010 0580	R/W (**Note**)	336
CRYP0_PE_PR_DTC	PE Particle Descriptor Destination Control	0x0 E010 0584	R/W (**Note**)	336

Note: Although this is a R/W register, in normal operation, the processor does not need to read or write it. Writing is not recommended since it can interfere with correct gather processing. It should be used for debug only.

Table 17-3. Security Function Registers (continued)

Register Mnemonic	Description	Address	Access	Page
Cryptographic Context (SA) Registers				
CRYP0_SA_CMD_0	SA Command 0 Word	0x0 E011 0600	Write only	338
CRYP0_SA_CMD_1	SA Command 1 Word	0x0 E011 0604	Write only	342
CRYP0_SA_KEY1_L	SA Key 1 Low	0x0 E011 0610	Write only	346
CRYP0_SA_KEY1_H	SA Key 1 High	0x0 E011 0614	Write only	346
CRYP0_SA_KEY2_L	SA Key 2 Low	0x0 E011 0618	Write only	346
CRYP0_SA_KEY2_H	SA Key 2 High	0x0 E011 061C	Write only	346
CRYP0_SA_KEY3_L	SA Key 3 Low	0x0 E011 0620	Write only	346
CRYP0_SA_KEY3_H	SA Key 3 High	0x0 E011 0624	Write only	346
CRYP0_SA_KEY4_L	SA Key 4 Low	0x0 E011 0628	Write only	346
CRYP0_SA_KEY4_H	SA Key 4 High	0x0 E011 062C	Write only	346
CRYP0_SA_IH_D0	SA Inner Hash Digest 0	0x0 E011 0630	Write only	347
CRYP0_SA_IH_D1	SA Inner Hash Digest 1	0x0 E011 0634	Write only	347
CRYP0_SA_IH_D2	SA Inner Hash Digest 2	0x0 E011 0638	Write only	347
CRYP0_SA_IH_D3	SA Inner Hash Digest 3	0x0 E011 063C	Write only	347
CRYP0_SA_IH_D4	SA Inner Hash Digest 4	0x0 E011 0640	Write only	347
CRYP0_SA_OH_D0	SA Outer Hash Digest 0	0x0 E011 0644	Write only	348
CRYP0_SA_OH_D1	SA Outer Hash Digest 1	0x0 E011 0648	Write only	348
CRYP0_SA_OH_D2	SA Outer Hash Digest 2	0x0 E011 064C	Write only	348
CRYP0_SA_OH_D3	SA Outer Hash Digest 3	0x0 E011 0650	Write only	348
CRYP0_SA_OH_D4	SA Outer Hash Digest 4	0x0 E011 0654	Write only	348
CRYP0_SA_SPI	SA IPsec SPI	0x0 E011 0658	R/W	349
CRYP0_SA_SEQ	SA IPsec Sequence Number	0x0 E011 065C	R/W	350
CRYP0_SA_SEQMKL	SA IPsec Sequence Number Mask Low	0x0 E011 0660	R/W	351
CRYP0_SA_SEQMKH	SA IPsec Sequence Number Mask High	0x0 E011 0664	R/W	351
CRYP0_SA_NONCE	SA Nonce Value	0x0 E011 0668	R/W	351
CRYP0_SA_PNTR	SA Pointer	0x0 E011 066C	R/W	352
CRYP0_SA_ARC4IJ	SA ARC4 i and j Pointer	0x0 E011 0670	R/W	352
CRYP0_SA_ARC4SB	SA ARC4 State Address Pointer	0x0 E011 0674	R/W	353
CRYP0_SA_IV_0	SA Initialization Vector 0	0x0 E011 06C0	R/W	353
CRYP0_SA_IV_1	SA Initialization Vector 1	0x0 E011 06C4	R/W	353
CRYP0_SA_IV_2	SA Initialization Vector 2	0x0 E011 06C8	R/W	353
CRYP0_SA_IV_3	SA Initialization Vector 3	0x0 E011 06CC	R/W	353
CRYP0_SA_HASH_B	SA Hash Byte Count	0x0 E011 01D0	R/W	354

Table 17-3. Security Function Registers (continued)

Register Mnemonic	Description	Address	Access	Page
CRYP0_SA_IH_0	SA Inner Hash 0 (mirror of CRYP0_SA_IH_D0)	0x0 E011 06D4	R(master and slave) W (master only	354
CRYP0_SA_IH_1	SA Inner Hash 1 (mirror of CRYP0_SA_IH_D1)	0x0 E011 06D8	R(master and slave) W (master only	354
CRYP0_SA_IH_2	SA Inner Hash 2 (mirror of CRYP0_SA_IH_D2)	0x0 E011 06DC	R(master and slave) W (master only	354
CRYP0_SA_IH_3	SA Inner Hash 3 (mirror of CRYP0_SA_IH_D3)	0x0 E011 06E0	R(master and slave) W (master only	354
CRYP0_SA_IH_4	SA Inner Hash 4 (mirror of CRYP0_SA_IH_D4)	0x0 E011 06E4	R(master and slave) W (master only	354
CRYP0_SA_ICV_0	SA ICV 0—HMAC result (outbound and inbound)	0x0 E011 06E8	Read only	355
CRYP0_SA_ICV_1	SA ICV 1—HMAC result (outbound and inbound)	0x0 E011 06EC	Read only	355
CRYP0_SA_ICV_2	SA ICV 2—HMAC result (outbound and inbound)	0x0 E011 06F0	Read only	355
CRYP0_SA_ICV_3	SA ICV 3—HMAC result (outbound and inbound)	0x0 E011 06F4	Read only	355
CRYP0_SA_ICV_4	SA ICV 4—HMAC result (outbound and inbound)	0x0 E011 06F8	Read only	355

Table 17-3. Security Function Registers (continued)

Register Mnemonic	Description	Address	Access	Page
True Random Number Generator (TRNG) Registers				
CRYP0_TRNG_DATA	TRNG Output	0x0 E012 0100	R/W	356
CRYP0_TRNG_STAT	TRNG Status	0x0 E012 0104	Read only	356
CRYP0_TRNG_CTRL	TRNG Test Control	0x0 E012 0108	R/W	357
CRYP0_TRNG_ENTA	TRNG Test Entropy A	0x0 E012 010C	R/W	358
CRYP0_TRNG_ENTB	TRNG Test Entropy B	0x0 E012 0110	R/W	359
CRYP0_TRNG_X0	TRNG Test Seed 0	0x0 E012 0114	R/W	359
CRYP0_TRNG_X1	TRNG Test Seed 1	0x0 E012 0118	R/W	359
CRYP0_TRNG_X2	TRNG Test Seed 2	0x0 E012 011C	R/W	359
CRYP0_TRNG_CNTR	TRNG Counter	0x0 E012 0120	R/W	360
CRYP0_TRNG_ALRM	TRNG Alarm Count	0x0 E012 0124	Read only	360
CRYP0_TRNG_CFG	TRNG Configuration	0x0 E012 0128	R/W	361
CRYP0_TRNG_LF0L	TRNG Test Read of LFSR 0 Low	0x0 E012 012C	Read only	362
CRYP0_TRNG_LF0H	TRNG Test Read of LFSR0 High	0x0 E012 0130	Read only	362
CRYP0_TRNG_LF1L	TRNG Test Read of LFSR1 Low	0x0 E012 0134	Read only	363
CRYP0_TRNG_LF1H	TRNG Test Read of LFSR1 High	0x0 E012 0138	Read only	363
CRYP0_TRNG_K0_L	TRNG Triple DES Key 0 Low	0x0 E012 013C	Write only	363
CRYP0_TRNG_K0_H	TRNG Triple DES Key 0 High	0x0 E012 0140	Write only	363
CRYP0_TRNG_K1_L	TRNG Triple DES Key 1 Low	0x0 E012 0144	Write only	364
CRYP0_TRNG_K1_H	TRNG Triple DES Key 1 High	0x0 E012 0148	Write only	364
CRYP0_TRNG_IV_L	TRNG Initialization Vector Low	0x0 E012 014C	Write only	364
CRYP0_TRNG_IV_H	TRNG Initialization Vector High	0x0 E012 0150	Write only	364
Public Key Accelerator (PKA) Registers				
CRYP0_PKA_A_PTR	PKA A Vector Address	0x0 E013 0800	R/W	365
CRYP0_PKA_B_PTR	PKA B Vector Address	0x0 E013 0804	R/W	365
CRYP0_PKA_C_PTR	PKA C Vector Address	0x0 E013 0808	R/W	365
CRYP0_PKA_D_PTR	PKA D Vector Address	0x0 E013 080C	R/W	365
CRYP0_PKA_A_LEN	PKA A Vector Length	0x0 E013 0810	R/W	365
CRYP0_PKA_B_LEN	PKA B Vector Length	0x0 E013 0814	R/W	365
CRYP0_PKA_SHIFT	PKA Shift	0x0 E013 0818	R/W	366
CRYP0_PKA_FUNC	PKA Function Code	0x0 E013 081C	R/W	367
CRYP0_PKA_COMP	PKA Comparison Result	0x0 E013 0820	Read only	370
CRYP0_PKA_DIV	PKA Address of Quotient MSW	0x0 E013 0824	Read only	371
CRYP0_PKA_MOD	PKA Address of Remainder MSW	0x0 E013 0828	Read only	372

Table 17-3. Security Function Registers (continued)

Register Mnemonic	Description	Address	Access	Page
Interrupt Controller Registers				
CRYP0_INT_UNMSK	Interrupt Unmasked Status	0x0 E015 00A0	Read only	374
CRYP0_INT_MSK	Interrupt Masked Status	0x0 E015 00A4	R/W	374
CRYP0_INT_EN	Interrupt Mask	0x0 E015 00A8	R/W	375
CRYP0_INT_CFG	Interrupt Configuration	0x0 E015 00AC	R/W	376
CRYP0_INT_DESRD	Interrupt Force Descriptor Read	0x0 E015 00B0	Write only	377
CRYP0_INT_DESCT	Interrupt Descriptor Count	0x0 E015 00B4	R/W	377
Device Controller Registers				
CRYP0_DC_CTRL	Device Control	0x0 E016 0080	R/W	378
CRYP0_DC_DEVID	Device ID	0x0 E016 0084	Read only	378
CRYP0_DC_DEVINF	Device Information	0x0 E016 0088	Read only	379
DMA Controller Registers				
CRYP0_DMA_USRC	DMA Source Address	0x0 E016 0094	R/W	380
CRYP0_DMA_UDST	DMA Destination Address	0x0 E016 0098	R/W	380
CRYP0_DMA_UCMD	DMA Command	0x0 E016 009C	R/W	381
CRYP0_DMA_CFG	DMA Configuration/Status	0x0 E016 00D4	R/W	382
Pseudo Random Number Generator (PRNG) Registers				
CRYP0_PRNG_STAT	PRNG Status	0x0 E017 0000	Read only	383
CRYP0_PRNG_CTRL	PRNG Control	0x0 E017 0004	R/W	384
CRYP0_PRNG_SDL	PRNG Seed Value Low	0x0 E017 0008	Write only	385
CRYP0_PRNG_SDH	PRNG Seed Value High	0x0 E017 000C	Write only	385
CRYP0_PRNG_K0L	PRNG Key 0 Low	0x0 E017 0010	Write only	385
CRYP0_PRNG_K0H	PRNG Key 0 High	0x0 E017 0014	Write only	385
CRYP0_PRNG_K1L	PRNG Key 1 Low	0x0 E017 0018	Write only	385
CRYP0_PRNG_K1H	PRNG Key 1 High	0x0 E017 001C	Write only	385
CRYP0_PRNG_RS0	PRNG Result 0 (31:0)	0x0 E017 0020	Read only	386
CRYP0_PRNG_RS1	PRNG Result 1 (63:32)	0x0 E017 0024	Read only	386
CRYP0_PRNG_RS2	PRNG Result 2 (95:64)	0x0 E017 0028	Read only	386
CRYP0_PRNG_RS3	PRNG Result 3 (127:96)	0x0 E017 002C	Read only	386
CRYP0_PRNG_LFL	PRNG LFSR Low	0x0 E017 0030	R/W	386
CRYP0_PRNG_LFH	PRNG LFSR High	0x0 E017 0034	R/W	386

17.2.3 CRYP0 SDR Control Register (SDR0_CRYP0)

The DESC command starts the Packet Engine state machine.

Address: SDR offset 0x4500

Size (bits): 32

Access: R/W

Reset: 0x0000 1C02

Figure 17-2. CRYP0 SDR Control Register (SDR0_CRYP0)

Bit	Mnemonic	Description	Comments
0	INT	DESC Command 0 No action 1 Initiate descriptor fetches in the Packet Engine	This bit must be cleared to 0 after being used.
1:18		Reserved	
19:31	AMASK	CRYP0 Base Address Base address of the Security Function.	Should always keep its reset value.

17.2.4 PE Control/Status Register (CRYP0_PE_CTLST)

The PE Control/Status Register provides basic command information to the PE. Together with the data pointed to in the SA structure, this provides the PE its instructions for processing a packet. Once the requested operation has completed, successfully or unsuccessfully, this register provides result status. Using this register, the processor can indicate whether the SA of the previous packet is the same as the SA from this packet. If that is the same, the SA does not have to be loaded. The current SA that is already available can be used to process this packet.

Address: 0x0 E010 0000

Size (bits): 32

Access: R/W

Reset: 0x0000 0042

Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST)

Bit	Mnemonic	Description	Comments
31:24	PCPS	Pad Control/Pad Status This field is used to specify Padding Control and return Padding Status. **Pad Control:** 0000 0000 No padding[1]. Otherwise, align packet end to modulo: 0000 0010 4-byte boundary 0000 0X00 8-byte boundary 0000 1000 16-byte boundary 0001 0000 32-byte boundary 0010 0000 64-byte boundary 0100 0000 128-byte boundary 1000 0000 256-byte boundary **Note 1:** For Zero Pad or Constant Pad mode no bytes are inserted. For PKCS#7 Pad, a pad-length field of one byte is inserted with a value of zero. For IPSEC Pad, a pad-length field of one byte followed by a Next Header field of one byte is inserted. For IPsec ESP outbound operations the packet is always padded to a 4-byte boundary. **Pad Status:** See comments.	**Pad Control:** Allows the Host to specify a Pad boundary for outbound operations. This feature may be used for "traffic flow security" in order to conceal the number of payload bytes in an encrypted packet. This field is ignored for all inbound operations. Basic outbound operations that use Stream Ciphers (ARC4 or AES Counter modes) when Pad Stream Cipher is disabled (CRYP0_SA_CMD_0[17] = 0), and SSL Outbound operations that use Stream Ciphers. **Pad Status:** This field is updated by the security function. For an inbound operation that uses IPsec, PKCS#7, SSL, or TLS Pad modes it returns the number of detected pad bytes. For all other inbound operations it returns zero since the other pad modes do not allow implicit determination of pad count. For an outbound operation it returns the number of inserted pad bytes for all Pad modes. In case of a Pad Verify Failure it returns zero. The Pad Count includes added bytes such as the pad length and Next Header field in an IPsec ESP pad.

Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST) (continued)

Bit	Mnemonic	Description	Comments
23:16	S	Status This byte encodes the result status for the operation requested in the corresponding Packet Descriptor. This is a read-only field in the Result Descriptor (it is "don't care" in the Command Descriptor). A write to any of the bits has no effect. The security function has built-in circuitry to detect faults that generate a packet failure condition. If the security function detects a fault, it reports the status in this byte to avoid having corrupted data enter the rest of the system.	
		0000 0000 No error. Successful completion	The packet is fully processed.
		xxxx xxx1 Authentication Failure For an inbound IPsec operation, the Integrity Check Value (ICV) did not match the computed value. For an inbound SSL or TLS operation, the Message Authentication Code (MAC) did not match the computed value.	The packet is fully processed. The SA is not updated for IPsec operations with header processing enabled, SSL operations and TLS operations.
		xxxx xx1x Pad Verify Failure For an inbound operation using IPsec, PKCS#7 or TLS padding, the decrypted pad did not match the expected values for the selected pad mode.	The packet is fully processed. The SA is not updated for IPsec operations with header processing enabled, TLS operations and SSL operations with TLS padding mode.
		xxxx x1xx Sequence Number Failure On an inbound IPsec operation, it indicates there was a fault in the Anti-Replay Sequence Number. On an outbound IPsec packet, it indicates a sequence number overflow (count has reached 2^{32} and has incremented to 0). The counter will wrap.	The packet is fully processed.
		0000 1xxx Invalid Command. Illegal packet command setting. Valid commands are: IPSEC, SSL, TLS, SRTP, Basic-Encrypt, Basic-Decrypt, Basic Hash, Encrypt-Hash, Hash-Decrypt. All other commands are invalid.	Packet command is ignored and no packet is processed. The packet must be re-queued or discarded.
		0001 1xxx Invalid Algorithm An invalid algorithm setting was selected in the SA record for Hash or Symmetric Crypto. Valid algorithms are: DES, Triple-DES, ARC4, AES, Null-Crypto, MD5, SHA-1 and Null-Hash. All other algorithms are invalid.	Packet command is ignored and no packet is processed. The packet must be re-queued or discarded.
		0010 1xxx Prohibited Algorithm A prohibited algorithm was selected in the SA record.Prohibited algorithms are: Triple-DES when Triple-DES is disabled, Basic operations with Null-Hash or Null-Crypto, SSL or TLS with Null-Hash, IPsec ESP with Null-Hash and Null-Crypto and IPsec with ARC4.	Packet command is ignored and no packet is processed. The packet must be re-queued or discarded.

Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST) (continued)

Bit	Mnemonic	Description	Comments
23:16 (cont.)	S	0011 1xxx Zero Length. The Packet Length field is zero, which is illegal. For IPsec the ICV is stripped before the length is checked. Any input length < 13 bytes results in an Zero Length Error.	Packet command is ignored and no packet is processed. The packet must be re-queued or discarded.
		0100 1xxx DMA Error A fatal error occurred during a DMA operation. A PLB timeout response occurred on a SA-Record read/write, SA-State read/write, ARC4 State read/write, or Packet Data read/write operation.	This packet is aborted. The packet must be re-queued or discarded. An Interrupt is generated. Scatter ring is reset.
		0101 1xxx Invalid IP Header Incorrect Packet Header for IPsec. Incorrect IP Headers are: - Headers that contain an invalid length in the length-field of the header preceding the AH. - Headers that contain an invalid next header field. - The AH is missing in the packet. - The input data of the AH/ESP packet is less than the IP Header + Extension Headers + AH/ESP Headers + IV, if applicable/available for the protocol. For IPsec the ICV is stripped before the length is checked.	This packet is aborted. The packet must be re-queued or discarded. An Interrupt is generated. The Scatter ring is reset.
		0110 1xxx IPsec SPI Mismatch On an inbound packet the 32-bit SPI value in the packet does not match the value in the SA while header processing is enabled.	The packet is fully processed.
		0111 1xxx Block Size Error The length of the inbound packet is not a multiple of the DES or AES block cipher length. For outbound packets the size is always automatically aligned to the correct block size. The hashed packet length is not a multiple of the hash block size in case of an intermediate hash operation. In case of a final hash operation, no error is generated. For IPsec the ICV is stripped before the Block Size is checked.	Packet is fully processed. An Interrupt is generated. Scatter ring is reset.

Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST) (continued)

Bit	Mnemonic	Description	Comments
23:16 (cont.)	S	1000 1xxx Escape Error Illegal combination of descriptor and/ or command settings is selected which can cause the PE to hang. Illegal combinations are: - IPSEC ESP with crypto algorithm ARC4. - IPSEC AH with Null-Hash and Copy_Header and Copy_Payload both zero. - SRTP with hash algorithm other than SHA-1. - SRTP with crypto algorithm other than null-crypto or AES-ICM or AES-CTR. - SSL or TLS with load IV from input. - SSL or TLS with crypto algorithm AES-CTR or AES-ICM mode. - SSL with other pad modes than SSL or TLS padding (only for block ciphers!). 1001 1xxx Length Error Illegal length detected. Basic operations: Packet data length ≤ Hash/Encrypt Offset SRTP operations: Packet data length ≤ IV (opt.) + Bypass Offset + ROC SSL operations: Packet data length ≤ 11 TLS operations: Packet data length ≤ 13 For IPsec the ICV is stripped before the length is checked. Any input length <12 bytes or non-4-byte aligned lengths result in a Length Error. 101x 1xxx Reserved 11xx 1xxx Reserved	Packet command is ignored and no packet is processed. The packet must be re-queued or discarded. Packet command is ignored and no packet is processed. The packet must be re-queued or discarded.

Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST) (continued)

Bit	Mnemonic	Description	Comments
15:8	NHPV	Next Header/Pad Value This field is used to pass the Next Header value between the processor and the security function. **Command descriptor:** For ESP outbound operations, the processor must populate this field with the value that is to be inserted (as part of the ESP trailer) into the Next Header field of the innermost operation's header. For Basic outbound operations that use Constant Pad mode or Constant SSL Pad mode the processor must specify the fixed constant Pad Value in this field. For Basic outbound operations that use IPsec Pad mode the processor must specify the Next Header value in this field. For SSL outbound operations that use Constant SSL Pad mode the processor must specify the fixed constant Pad Value in this field. For all other outbound operations and all inbound operations this field is not used. **Result descriptor:** For IPsec ESP outbound operations this field returns the decimal value 50. For IPsec AH outbound operations this field returns the decimal value 51. For all other outbound operations the security function will not update this field. For IPsec inbound operations and Basic inbound operations that use IPsec padding mode the security function returns the Next Header field it detects on the innermost operation's header, which will typically be the value for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination options, another header value could be seen. For all other inbound operations and in case of an error the returned Pad Value is zero.	
7:6	CMD	SA BusID Selects the Bus ID where the SA-Record will be located.	In the security function, these bits are ignored (set to 01). These bits always return 01 on a read by the processor.
5	CMD	Use Cached SA 0 New SA must be fetched from processor memory 1 SA currently in the PE is used for this packet	When 1, the SA for this packet does not have to be reloaded. All fields used for the previous packet can be reused, including the updated fields. These updates correspond directly to the values that are necessary for this packet. Note that only the SA and the Saved IV of the State Record are cached in the PE. The Saved Inner Digest Registers and the Saved Hash Byte Count of the State Record are not cached. When the Save Hash Digest bit is set the Inner Digest value and Hash Byte Count are overwritten for the next operation with the original Inner Digest and Hash Byte Count value of the SA. The Use Cached SA bit must not be used for chained hash operations where information is passed through the SA State to the next operation. This is the case for all basic hash or protocol operations where the hash state must be saved. This bit always returns the last written value on a read by the processor.

Figure 17-3. PE Control/Status Register (CRYP0_PE_CTLST) (continued)

Bit	Mnemonic	Description	Comments
4	CMD	Hash Final 0 Specifies that the PE perform an intermediate hash operation by generating an intermediate hash digest on the data presented. 0 Specifies that the PE append the required final hash pad and generate the final hash digest on the data presented. This completes the hash operation.	This bit is only applicable for Basic Hash, Basic Encrypt-Hash and Basic Hash-Decrypt operations and is not required for IPsec, SSL, TLS or SRTP protocol operations. This bit always returns the last written value on a read by the processor.
3	CMD	Init ARC4 Initialize the ARC4 cryptographic algorithm. 0 The ARC4 initialization depends on the value of CRYP0_SA_COMMAND1 bit #29, ARC Stateless / Stateful mode. If this bit is set to 1, the ARC4 State Record and i/j pointer are loaded to continue the encrypt/decrypt processing from the previous algorithm state. If this bit is set to 0, the key is read from the SA-record, and the ARC4 S-boxes are initialized using this key, prior to the encryption/decryption of data. 1 Specifies that this is the first packet to be processed with a new key. The key is read from the SA-record, and the ARC4 S-boxes are initialized using this key, prior to the encryption/decryption of data.	This bit is only applicable for ARC4 mode.
2		Reserved	
1	CMD	CryptCore Done 0 The security function has not finished processing this descriptor (or Command Queue) 1 The security function has finished processing this descriptor (or Command Queue) and has returned ownership to the processor.	This bit can be reset by both the Host and the security function, but only the security function can set this bit. This bit always returns 1 on a read by the processor.
0	CMD	Host Ready 0 Processor has not populated the descriptor. 1 Processor has populated the descriptor. The processor function assumes that it can begin processing the descriptor.	This bit can be reset by both the processor and the security function, but only the processor can set this bit. This bit always returns 0 on a read by the processor. Note that this bit is only read by the security function in Autonomous Ring mode; it is ignored when directly writing to the Command Queue (instead, the command is fired by the write of the last word in the Command Queue).

17.2.5 PE Source Address Register (CRYP0_PE_SOURCE)

This register specifies the starting address for the packet to be processed.

Address: 0x0 E010 0004

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-4. PE Source Address Register (CRYP0_PE_SOURCE)

Bit	Mnemonic	Description	Comments
31:0		Packet Source Addess	This address does not have to be on a word boundary. For Gather processing, this register contains a pointer to the first Gather Particle Descriptor (which then points to the first buffer).

17.2.6 PE Destination Address Register (CRYP0_PE_DEST)

This register specifies the starting address at which to write the result data from the requested operation.

Address: 0x0 E010 0008

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-5. PE Destination Address Register (CRYP0_PE_DEST)

Bit	Mnemonic	Description	Comments
31:0		Packet Destination Addess	This address does not have to be on a word boundary. For Scatter processing, the Packet Destination Address is ignored on input. Upon completion of packet processing, this field is updated in the Result Descriptor with a pointer to the first Scatter Particle Descriptor (which then points to the first buffer).

17.2.7 PE SA Address Register (CRYP0_PE_SA)

This register specifies the starting address of the SA record.

Address: 0x0 E010 000C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 17-6. PE SA Address Register (CRYP0_PE_SA)

Bit	Mnemonic	Description	Comments
31:0		SA Addess	Generally, this address is on a word boundary. However, it is not mandatory. This register is not used when Use Cached SA is specified.

17.2.8 PE Length Register (CRYP0_PE_LENGTH)

When performing direct writes to the Command Queue (Descriptor Ring disabled), a write to this register will "fire" the command into the PE. This takes the place of the ownership bits in the command byte in the Descriptor Ring modes. Since this word is the last in the Command Queue, it should be written last, after the other registers have been properly programmed.

Address: 0x0 E010 0010

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-7. PE Length Register (CRYP0_PE_LENGTH)

Bit	Mnemonic	Description	Comments
31:24	BYP	Bypass Specifies the offset, in words, between the hash data and the encryption/decryption data for SRTP operations.	The Hash Encrypt Offset in CRYP0_SA_CMD_1 has the same function, except that it is part of the SA and, therefore, applies the same value for all packets. Since for SRTP the Hash Encrypt Offset can be variable on a per packet basis, it is part of the descriptor.
23	CCD	CryptCore Done Mirrors the equivalent bit in the first Control/ Status word of the descriptor.	Repeated in the last word in order to guarantee ownership consistency between the first and last word of the descriptor. When the descriptor fetch engine reads a descriptor, this ownership bit must match that in the first word, or the descriptor is discarded and fetched again. A write to this bit has no effect.
22	HRDY	Host Ready Mirrors the equivalent bit in the first Control/ Status word of the descriptor.	Repeated in the last word in order to guarantee ownership consistency between the first and last word of the descriptor. When the descriptor fetch engine reads a descriptor, this ownership bit must match that in the first word, or the descriptor is discarded and fetched again. A write to this bit has no effect.
21:20		Reserved	
19:0	L	Length Total length (in bytes) of all data passed to the Input Buffer of the security function for an operation. Valid length range is 1 to 1,048,575 bytes.	A length of 0 bytes is illegal and will result in an error status code in the Result Descriptor. In case of a DMA Error, Block Size Error or Invalid IP Header Error the Length field returns zero in the Result Descriptor. These bits always return the last written value on a read by the processor.

17.2.9 PE DMA Configuration Register (CRYP0_PE_DMA_CF)

This register is used to select static settings that control the packet-processing path. These settings are typically set at initialization and not changed again.

Address: 0x0 E010 0040

Size (bits): 32

Access: See register fields

Reset: 0x0014 5050

Figure 17-8. PE DMA Configuration Register (CRYP0_PE_DMA_CF)

Bit	Mnemonic	Description	Comments
31:25		Reserved	
24	DHDMAM	Direct Host DMA Mode Determines security function mode. 0 Master Mode 1 Direct Host DMA Mode	In Direct Host DMA Mode (mastering ability is disabled), the processor must write all data to the security function.
23:21		Reserved	
20	SPDROU	Suppress PDR Ownership Update Determines whether the security function updates the ownership bits in the Command Descriptor on the PDR. 0 Clear the ownership bits of the Command Descriptor when it finishes an operation. This prevents the security function from reprocessing an old descriptor when it wraps around the PDR. 1 Do not clear the ownership bits in the packet descriptor when it completes an operation. In this case, the processor is responsible for clearing the ownership bits.	This setting is ignored if the PDR and RDR overlap. The processor must clear these ownership bits before the security function is allowed to wrap entirely around the PDR to reencounter old descriptors. As long as there is at least one non-security function-owned descriptor separating the newest valid descriptor and the oldest, this does not occur. Choosing this setting has the advantage of eliminating a separate DMA write to the PDR. The Result Descriptor is always written by the security function.
19:16		Reserved	
15:14	SPRBID	Scatter Particle Ring Bus ID Selects the bus ID where the Scatter Particle Ring is located. These bits are always set to 01.	This is a Read only field.
13:12	GPRBID	Gather Particle Ring Bus ID Selects the bus ID where the Gather Particle Ring is located. These bits are always set to 01.	This is a Read only field.
11		Reserved	

Figure 17-8. PE DMA Configuration Register (CRYP0_PE_DMA_CF) (continued)

Bit	Mnemonic	Description	Comments
10	PFDES	Packet Follows Descriptor This bit along with SAPP are used as a 2-bit Descriptor Ring mode selector. They set-up the way that the PE fetches descriptors, SAs, and packet data. 00 Basic Ring Mode - Descriptor, SA, and packet data are located in different memory locations. The descriptor is located in a separate descriptor ring with pointers to input/output packet data and SA record. 01 SA Packet Mode - SA and packet data are located successively in memory. Only pointers for input and output are provided. 10 Packet Descriptor Packet Mode - Descriptor and packet data are located successively in memory. The complete data blocks (packet descriptor + packet data) have a fixed size. Only an SA record pointer is required. 11 Packet Descriptor SA Packet Mode - All Descriptor, SA, and packet data are located successively in memory. The complete data blocks (packet descriptor + SA + packet data) have a fized size. No pointers are required	
9	SAPP	SA Precedes Packet See PFDES bit above.	
8	PDRE	PDR Enable Selects how the security function receives commands for the PE. 0 Disable the Packet Descriptor Ring manager in the security function and PE commands must be individually entered 1 Enable the Packet Descriptor Ring (PDR) Engine and fetch PE commands from the Descriptor Ring	In Direct Host DMA mode this bit is forced to 0.
7:3		Reserved	
2	RSCG	Reset Scatter/Gather Cache Controls the reset to the Scatter/Gather state machine 0 Release the Scatter/Gather Cache reset 1 Reset the Scatter/Gather Cache	Resets the pointers to the Particle Descriptors to their base address setting. This reset must be coordinated with the owner'of the Particle buffers to ensure that the pointers are synchronized after the reset. For a momentary reset, the processor can set this bit to 1 and back to 0 again with no delay between the sets.
1	RPDRCP	Reset PDR Counters/Pointers Controls reset to the Packet Descriptor Ring state machine. 0 Release the Packet Descriptor Ring state machine reset 1 Reset the Packet Descriptor Ring state machine	Resets the pointers for both the Packet Descriptors and Result Descriptors to their base address setting. If the PDR ring is not enabled, this bit may be left in the reset state. This reset must be coordinated with the owner of the Descriptor Ring to ensure that the pointers are synchronized after the reset. For a momentary reset, the processor can set this bit to 1 and back to 0 again with no delay between the sets.

Figure 17-8. PE DMA Configuration Register (CRYP0_PE_DMA_CF) (continued)

Bit	Mnemonic	Description	Comments
0	RPE	Reset PE Controls reset to both the PE and the state machine logic that drives header processing, the DMA Controller, and context management. 0 Release the PE reset 1 Reset the PE	For a momentary reset, the processor can set this bit to 1 and back to 0 again with no delay between the sets.

17.2.10 PE DMA Status Register (CRYP0_PE_DMA_ST)

Provides the status of the PE.

Address: 0x0 E010 0044

Size (bits): 32

Access: Read only

Reset: 0x0200 0C03

Figure 17-9. PE DMA Status Register (CRYP0_PE_DMA_ST)

Bit	Mnemonic	Description	Comments
31:22	SEOS	CryptCore Output Size Number of 32-bit words that the PE is requesting be read from the output buffer.	
21:12	SEIS	CryptCore Input Size Number of 32-bit words are available in the PE input buffer.	
11	CQS	Output Request Active Active-low (0) indicates that the PE is requesting that output data be read.	Mirrors the $\overline{\text{OutputRequest}}$ signal that acts as a DMA output request in Direct Host DMA mode.
10		Input Request Active Active-low (0) indicates that the PE is requesting that input data be written.	Mirrors the $\overline{\text{InputRequest}}$ signal that acts as a DMA input request in Direct Host DMA mode.
9		Command Queue Active Indicates that the PE is currently processing a packet.	This bit is 0 when the engine is idle.
8		Reserved	

Figure 17-9. PE DMA Status Register (CRYP0_PE_DMA_ST) (continued)

Bit	Mnemonic	Description	Comments
7	CQS	SPI Mismatch Inbound SPI did not match the expected value supplied in the SA.	
6		ICV Fault Inbound ICV fault detected	The ICV carried within the packet did not match the value just computed.
5		Crypto Pad Fault Inbound cryptographic pad fault detected	Possible faults include Pad Count field not matching the number of Pad bytes detected and Pad field values not matching specified pattern.
4		Outer Hash Done Outer hash processing for this packet is finished	
3		Inner Hash Done Inner hash processing for this packet is finished	
2		Encryption Done Encryption or decryption for this packet is finished	
1		CryptCore Output Done All of the output data for the current packet has been read from the cryptographic output buffer.	
0		CryptCore Input Done Number of bytes specified in the packet descriptor has been written into the cryptographic input buffer.	

17.2.11 PE Packet Descriptor Ring Base Address Register (CRYP0_PE_PDR_BA)

Normally this register contains the Packet Descriptor Ring Base Address (*Figure 17-10*). For diagnostic purposes it can contain the Crypto Length Out (*Figure 17-11*).

Address: 0x0 E010 0048

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-10. PE Packet Descriptor Ring Base Address Register (CRYP0_PE_PDR_BA)

Bit	Mnemonic	Description	Comments
31:0		Packet Descriptor Ring Base Address Allows the processor to specify the base location of the Packet Descriptor Ring in external memory space.	This register is ignored if the CRYP0_PE_RING_S register is set to 0 (indicating that the processor writes directly into descriptor ring).

OR

If in Direct Host DMA mode for read access:

Figure 17-11. PE Packet Descriptor Ring Base Address Register (CRYP0_PE_PDR_BA)

Bit	Mnemonic	Description	Comments
31:20		Reserved	
19:0	CLO	Crypto Length Out Returns the length of the encrypted part of the packet in the PE Output Buffer in bytes.	The encrypted part is the total packet minus the ICV (for IPsec) or TAG (for SRTP). This field should be used for debugging purposes only.

17.2.12 PE Packet Result Ring Base Address Register (CRYP0_PE_RDR_BA)

This register contains the Packet Result Ring Base Address.

Address: 0x0 E010 004C

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-12. PE Packet Result Ring Base Address Register (CRYP0_PE_RDR_BA)

Bit	Mnemonic	Description	Comments
31:0		Packet Result Ring Base Address Allows the processor to specify the base location of the Packet Result Ring in external memory space.	This register is applicable even if the CRYP0_PE_RING_S register is set to 0 (indicating that the Host will write directly into command register set). Result Descriptors are always written to a specified ring location. They are never written to the internal Command Queue.

17.2.13 PE Ring Size and Offset Register (CRYP0_PE_RING_S)

This register contains the PE Descriptor Ring Size and Offset.

Address: 0x0 E010 0050

Size (bits): 32

Access: R/W

Reset: 0x0005 0000

Figure 17-13. PE Ring Size and Offset Register (CRYP0_PE_RING_S)

Bit	Mnemonic	Description	Comments
31:16	DRO	Descriptor Ring Offset Offset (in words) between Descriptor Ring entries for an external descriptor ring	This value applies to both the PDR and the RDR. The Descriptor Ring Offset field is only applicable when CRYP0_PE_DMA_CF[PFDES] is 1. When PFDES is 0, the implicit offset must be fixed at five words. This situation occurs after a reset. Settings of 5–65535 represent valid external ring offsets. This offset must allow space for the Packet Descriptor (5W), the packet data, and, optionally, the SA-Record (32W if SAPP is 1). Typical values for the offset would range from 512W (2KB) to 8192W (32KB).
15:10		Reserved	
9:0	DRS	Descriptor Ring Size Descriptor Ring Size (number of entries) for an external descriptor ring	This value applies to both the PDR and the RDR. Settings of 1–1023 represent valid external ring sizes. A setting of 0x000 specifies that an external PDR is not used, and that the processor performs PLB Slave writes of descriptors directly into the Command Queue. This effectively turns off the Descriptor Fetch Engine. In this case, the Result Descriptor Ring is fixed at a size of 1.

17.2.14 PE Ring Poll Register (CRYP0_PE_RING_P)

This register allows programming of two polling parameters that are used in the Descriptor Fetch Engine. The first parameter is the basic polling frequency at which the security function reads a segment of the external PDR into its internal Command Queue.

The second parameter is the retry interval that specifies how long the security function should wait in between retry of reads on an invalid descriptor entry (ownership bit not yet assigned to the security function). The retry Interval should be set to a shorter period than the poll interval. Otherwise, the next poll event will preempt a retry. Both of these parameters are typically used to limit the amount of bus bandwidth that is consumed by the descriptor polling process.

A Read Descriptor interrupt can preempt the poll interval and cause a poll to occur sooner than the timer would dictate. Should this occur, the timer polling interval is reset to its starting point, creating a full delay before the next timed poll.

If descriptors are written directly into the internal Command Queue, this register is ignored. This mode is configured in CRYP0_PE_DMA_CF.

Address: 0x0 E010 0054

Size (bits): 32

Access: R/W

Reset: 0x8000 0000

Figure 17-14. PE Ring Poll Register (CRYP0_PE_RING_P)

Bit	Mnemonic	Description	Comments
31	C	Continuous Specifies that the Descriptor Fetch Engine should continuously read descriptor segments until the Command Queue has a valid descriptor	Once the Command Queue entry becomes free, the security function continuously polls for the next valid descriptor (although the polls will be interleaved with other DMA transfers such as moving packet data in and out of the security function).
30:26			
25:16	RRDIV	Ring Retry Divisor Binary value that is used to divide the main security function clock at the retry frequency.	Values of 1–1023 provide valid retry intervals. A setting of 0x000 disables retries altogether and the security function waits until the next Poll interval occurs. A fixed divisor of 128 is inserted in the main clock prior to the Ring Retry Divisor. **Example:** Main security function clock: 133 MHz /128 prescale = 1039 kHz Ring Poll set to 0x001: Frequency = 1039 kHz (0.96 µs between retries) Ring Poll set to 0x3FF: Frequency = 1016 Hz (985 µs between retries)
15:12			

Figure 17-14. PE Ring Poll Register (CRYP0_PE_RING_P) (continued)

Bit	Mnemonic	Description	Comments
11:0	RPDIV	Ring Poll Divisor Binary value used to divide the main security function clock at the polling frequency	Values of 1–4095 provide valid poll intervals. A setting of 0x000 disables PDR polling altogether. In this case, the Read Descriptor interrupt is the only mechanism for initiating a descriptor poll once an empty descriptor has been encountered. A fixed divisor of 128 is inserted in the main clock prior to the Ring Poll Divisor. **Example:** Main security function clock: 133 MHz /128 = 1039 kHz Ring Poll 0x001: Frequency = 1039 kHz (0.96 µs between polls) Ring Poll 0xFFF: Frequency = 254 Hz (3.94 ms between polls)

17.2.15 PE Internal Ring Status Register (CRYP0_PE_I_RING)

This register is used to provide real-time status for the internal Packet Descriptor Command Queue. In general, this register is used for debug purposes. But in the case of Command Descriptors being written directly to the Command Queue, it is used to determine how full the Command Queue is.

Address: 0x0 E010 0058

Size (bits): 32

Access: Read only

Reset: 0x0000 0001

Figure 17-15. PE Internal Ring Status Register (CRYP0_PE_I_RING)

Bit	Mnemonic	Description	Comments
31:1		Reserved	
0	CQ0A	Command Queue 0 Available 0 Queue not available 1 Queue available for a PE command	In the Target Command mode, CQ0A = 1 indicates that the user can write a new packet command into this register.

17.2.16 PE External Ring Status Register (CRYP0_PE_E_RING)

This register is used to provide real-time status for the external PDR (if applicable).

Address: 0x0 E010 005C

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-16. PE External Ring Status Register (CRYP0_PE_I_RING)

Bit	Mnemonic	Description	Comments
31:26		Reserved	
25:16	ONPD	Offset of Next Packet Descriptor	Indicates the offset from the address in CRYP0_PE_PDR_BA of the Next Packet Descriptor to be read by the PE.
15:0		Reserved	

17.2.17 PE I/O Threshold Register (CRYP0_PE_IO_THR)

This register is used to specify at what "high water" and "low water" points the PE should begin to transfer packet data into or out of the PE buffer RAM. These parameters are useful for controlling the DMA burst size for packet data input and output from the PE.

Address: 0x0 E010 0060

Size (bits): 32

Access: R/W

Reset: 0x0008 0008

Figure 17-17. PE I/O Threshold Register (CRYP0_PE_IO_THR)

Bit	Mnemonic	Description	Comments
31:26		Reserved	
25:16	PEOTH	Output Threshold Number of 32-bit words (1-446) that must be available in the PE Output Data RAM buffer prior to initiating a DMA output transfer	The maximum threshold is 446 instead of 511 bytes since this buffer is also used to store up to 256 pad bytes that can be stripped for decryption operations.
15:10		Reserved	
9:0	PEITH	Input Threshold Number of 32-bit words (1-508) that must be available in the PE Input Data RAM buffer prior to initiating a DMA input transfer	The maximum threshold is 508 instead of 511 bytes since this buffer is also used to redirect a 12-byte ICV for IPsec AH operations.

17.2.18 PE Gather Particle Ring Base Address Register (CRYP0_PE_GATH)

This register allows the processor to specify the base location of the Gather Particle Ring.

Address: 0x0 E010 0064

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-18. PE Gather Particle Ring Base Address Register (CRYP0_PE_GATH)

Bit	Mnemonic	Description	Comments
31:0		Gather Particle Ring Source Address	This register is ignored if gather is not enabled in any SA-record.

17.2.19 PE Scatter Particle Ring Base Address Register (CRYP0_PE_SCAT)

This register allows the processor to specify the base location of the Scatter Particle Ring.

Address: 0x0 E010 0068

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-19. PE Scatter Particle Ring Base Address Register (CRYP0_PE_SCAT)

Bit	Mnemonic	Description	Comments
31:0		Scatter Particle Ring Source Address	This register is ignored if scatter is not enabled in any SA-record.

17.2.20 PE Particle Ring Size Register (CRYP0_PE_PT_S)

This register contains the scatter and gather particle ring sizes.

Address: 0x0 E010 006C

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-20. PE Particle Ring Size Register (CRYP0_PE_PT_S)

Bit	Mnemonic	Description	Comments
31:16	SPDRS	Scatter Particle Descriptor Ring Size Number of 2-word entries	Valid sizes range is from 2 to 65532.
15:0	GPDRS	Gather Particle Descriptor Ring Size Number of 2-word entries	Valid sizes range is from 2 to 65532.

17.2.21 PE Particle Ring Configuration Register (CRYP0_PE_PT_CFG)

This register is used to select static settings that control scatter/gather processing. These settings are typically set at initialization and not changed again.

Address: 0x0 E010 0070

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-21. PE Particle Ring Configuration Register (CRYP0_PE_PT_CFG)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15:0	SPS	Scatter Particle Size Size, in bytes, of each particle in the scatter ring	Valid sizes range is from 4 to 65532 bytes in multiples of four bytes. A non-zero value must be written.

17.2.22 PE Particle Descriptor Source Address Register (CRYP0_PE_PR_SCA)

This register contains the particle descriptor source address. Although this is a read/write register, in normal operation, the processor does not read or write it. Writing is not recommended since it can interfere with correct gather processing. The register should be used for debug only.

Address: 0x0 E010 0500

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-22. PE Particle Descriptor Source Address Register (CRYP0_PE_PR_SCA)

Bit	Mnemonic	Description	Comments
31:0	SPS	Particle Address Starting address of the Gather Particle	Set by the processor in the External Particle Ring.

17.2.23 PE Particle Descriptor Source Control Register (CRYP0_PE_PR_SCC)

This register contains the particle descriptor source control. Although this is a read/write register, in normal operation, the processor does not read or write it. Writing is not recommended since it can interfere with correct gather processing. The register should be used for debug only.

Address: 0x0 E010 0504

Size (bits): 32

Access: R/W

Reset: 0x0000 0002

Figure 17-23. PE Particle Descriptor Source Control Register (CRYP0_PE_PR_SCC)

Bit	Mnemonic	Description	Comments
31:16	PS	Particle Size Size of the Scatter Particle	Set by the processor in the External Particle Ring. This has to be a value in the range of 4 to 65532 bytes in multiples of four bytes.
15:2		Reserved	
1	PEDONE	0 Security function has not finished processing the particle descriptor 1 Security function has finished processing the particle descriptor and has returned ownership to the processor	Written by the security function. Read by the processor. In conjuction with HDRDY (Bit 0): Bits 1 and 0, 00 = Unassigned. Security function ignores. 01 = Processor has populated particle data and particle descriptor. Security function now owns. 10 = Security function done. Ownership returned to the processor. 11 = Reserved
0	HDRDY	Host Ready 0 The processor has not populated the particle descriptor 1 The processor has populated the particle descriptor	Written by the processor. Read by the security function. In conjuction with PEDONE (Bit 1): Bits 1 and 0, 00 = Unassigned. Security function ignores. 01 = Processor has populated particle data and particle descriptor. Security function now owns. 10 = Security function done. Ownership returned to the processor. 11 = Reserved

17.2.24 PE Particle Destination Address Register (CRYP0_PE_PR_DTA)

This register contains starting address of the Scatter Particle, as set by the processor in the External Particle Ring. Although this is a read/write register, in normal operation, the processor does not read or write it. Writing is not recommended since it can interfere with correct gather processing. The register should be used for debug only.

Address: 0x0 E010 0580

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-24. PE Particle Destination Address Register (CRYP0_PE_PR_DTA)

Bit	Mnemonic	Description	Comments
31:0	SPS	Particle Address	Set by the processor in the External Particle Ring.

17.2.25 PE Particle Destination Control Register (CRYP0_PE_PR_DTC)

This register contains particle destination control information.

Address: 0x0 E010 0584

Size (bits): 32

Access: R/W

Reset: 0x0000 0002

Figure 17-25. PE Particle Destination Control Register (CRYP0_PE_PR_DTC)

Bit	Mnemonic	Description	Comments
31:2		Reserved	There is no particle size field in this register. The size of the Scatter Particle is programmable (but fixed and the same for all packets during packet processing). The processor can set by this size (4 to 65532 bytes) in the Particle Ring Configuration Register (CRYP0_PE_PT_CFG).
1	PEDONE	Security function done 0 Security function has not finished processing the particle descriptor 1 Security function has finished processing the particle descriptor and has returned ownership to the processor	Written by the security function. Read by the processor. In conjuction with HDRDY (Bit 0): Bits 1 and 0, 00 = Unassigned. Security function ignores. 01 = Processor has populated particle data and particle descriptor. Security function now owns. 10 = Security function done. Ownership returned to the processor. 11 = Reserved
0	HDRDY	Host Ready 0 The processor has not populated the particle descriptor 1 The processor has populated the particle descriptor	Written by the processor. Read by the security function. In conjuction with PEDONE (Bit 1): Bits 1 and 0, 00 = Unassigned. Security function ignores. 01 = Processor has populated particle data and particle descriptor. Security function now owns. 10 = Security function done. Ownership returned to the processor. 11 = Reserved

17.2.26 SA Command 0 Register (CRYP0_SA_CMD_0)

The Security Association (SA) record, along with the packet descriptor, provides the PE all of the necessary information to process an operation. The SA-record contains information which is either static for the lifetime of the association or is dynamically updated by the PE. Any control information which must be modified by the processor for each operation is contained in the descriptor.

CRYP0_SA_CMD_0 contains the major control bits for defining the cryptographic operation.

CRYP0_SA_CMD_0 is Write-only and always return zero on a read by the processor.

Address: 0x0 E011 0600

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-26. SA Command 0 Register (CRYP0_SA_CMD_0)

Bit	Mnemonic	Description	Comments
31	OSCAT	Output Scatter Controls whether output packet data is scattered 0 Output packet data is contiguous in memory 1 Output packet data must be scattered to memory	
30	IGATH	Input Gather Controls whether input packet data is gathered 0 Input packet data is contiguous in memory 1 Input packet data must be gathered from memory	
29	SHSTT	Save Hash State Controls saving the hash state after completion of an SHA-1, MD5 or HMAC operation. 0 Do not save the hash state to the State Record 1 Save the hash state to the State Record	The hash state includes the interim hash digest and the hash byte count.
28	SIV	Save IV Controls saving the IV to the State Record after completion of a DES/Triple-DES or AES operation. 0 Do not save the IV to the State Record 1 Save the IV to the State Record	This bit must be cleared for DES ECB mode or AES ECB mode, in case no IV is used.
27:26	LHSTT	Load Hash State Controls Hash loading for the SHA-1 and MD5 algorithms. 00 From SA or State (digest only, Hash Byte Count = 0x40) 01 Reserved 10 From state (read Saved Inner Hash Digest and Saved Hash Byte Count) 11 No load (use the Hash algorithm defined constants for the initial Hash. Hash Byte Count = 0x00)	

Figure 17-26. SA Command 0 Register (CRYP0_SA_CMD_0) (continued)

Bit	Mnemonic	Description	Comments
25:24	IVLD	IV Loading Controls IV loading for the AES, DES and Triple-DES cryptographic algorithms. For cryptographic algorithms that do not require an IV, these bits are ignored. 00 Reuse (no load) - not applicable for inbound data 01 From input buffer 10 From State record 11 From internal PRNG	
23:22	OBID	Output Bus ID Selects the bus ID where the output data will be located.	In the security function, these bits are ignored (fixed to 01). These bits always return 01 on a read by the processor
21:20	IBID	Input Bus ID Selects the bus ID where the input data will be located.	In the security function, these bits are ignored (fixed to 01). These bits always return 01 on a read by the processor
19	HP	Header Processing Controls how the security function performs header processing for this SA-record flow. This bit is only applicable for IPsec Protocol operations. There is no Header Processing for Basic Operations, SSL, TLS and SRTP protocol operation. However, for SSL Hash and TLS Hash operations the Header must be supplied to the PE. The same is applicable for SRTP Encryption-Hash operations. **IPSEC Outbound** (3:0 = 0000)—Determines whether the security function performs IPsec header insertion. 0 Processor subsystem is expected to insert either an AH or ESP header into the correct location within a packet. The security function inserts the ICV into the provided hole in the AH header. 1 Security function inserts an AH or ESP header in the proper location for an outbound packet, including the SPI and Sequence Counter. **IPSEC Inbound** (Bits 3:0 = 1000)—Determines whether the security function performs IPsec header verification. 0 Processor subsystem is expected to verify the AH or ESP header. 1 Security function verifies the AH or ESP header in the inbound packet, including the Sequence Number verification and Sequence mask processing	Fragment Headers processing is not supported by the security function. Any Fragment Header passed into the PE is ignored. So the Host *must* make sure that fragmentation headers are stripped before entering the packet in the security function. Should the Fragment Header be present in the packet, it is treated as an extension header (that is, it is ignored as being a Fragment Header, but not skipped over and the previous header's Next Header field is not changed to be the Next Header field in the Fragment Header). The length field of this header is fixed at 64 bits. For an IPsec IPv6 inbound packet any order of Extension Headers is allowed. The security function uses the following rules for placing an AH Header in an IPv6 outbound packet: 1. The AH Header is always inserted before the payload data where payload is defined as either the protocol data or the start of a tunneled packet. 2. If there is one Destination Header the AH Header will be inserted between the Destination Header and the payload. 3. If there are multiple Destination Headers before the payload then the AH header is inserted before the second Destination Header. A possible third Destination Header (although not allowed by the RFC 2402) is seen as payload.
18	EPAD	Extended Pad Extends the number of Pad options for the security function 0 Basic Pad modes 1 Extended Pad modes required for SSL and TLS protocol operations	See CPAD.
17	PSC	Pad Stream Ciphers Controls padding for Basic operations that use Stream Ciphers. 0 Disable padding 1 Enable padding	Protocols operations that use Stream Ciphers are always padded according to the protocol requirements.

Figure 17-26. SA Command 0 Register (CRYP0_SA_CMD_0) (continued)

Bit	Mnemonic	Description	Comments
16		Reserved	
15:12	HALGO	Hash Algorithm Selects the Hash algorithm. 0000 MD5 0001 SHA-1 0010 – 1110 Reserved 1111 Null	Not all hash algorithms are allowed for protocol operations.
11:8	CALGO	Crypto Algorithm Selects the Symmetric Encryption/Decryption algorithm. Not all algorithms are allowed for protocol operations. 0000 DES 0001 Triple-DES 0010 ARC4 0011 AES 0100 – 1110 Reserved 1111 Null	
7:6	CPAD	Crypto Pad Selects the method of cryptographic padding. If EPAD = 0: 00 IPsec 01 PKCS#7 10 Constant Pad 11 Zero If EPAD = 1: 00 Reserved 01 TLS 10 Constant SSL 11 Reserved	These bits are ignored for IPsec and TLS operations.
5:4	OGRP	Operation Group: These two bits select one of four groupings of operations: 00 Basic Operations 01 Reserved 10 Reserved 11 Protocol Operations	See tabxref1 and tabxref2.
3	IO	Inbound/Outbound processing 0 Outbound 1 Inbound	For example, the encryption and compression operations are considered Outbound, and decryption and decompression are considered Inbound.
2:0	OCODE	OpCode These three bits select a specific sub-operation from within the Operation Group.	See *Table 17-4* and *Table 17-5* below.

Table 17-4. Basic Operation Decoding

Outbound				Inbound			
Operation Group	**In/Out**	**OpCode**	**Description**	**Operation Group**	**In/Out**	**OpCode**	**Description**
00	0	000	Encryption	00	1	000	Decryption
00	0	001	Encryption-Hash	00	1	001	Hash-Decryption
00	0	010	Reserved (compress)	00	1	010	Reserved (Decompress)
00	0	011	Hash	00	1	011	Reserved (Hash)
00	0	100	Reserved (Hash-Encryption	00	1	100	Reserved Decryption-Hash)
00	0	101–111	Reserved	00	1	101–111	Reserved

Table 17-5. Protocol Operation Decoding

Outbound				Inbound			
Operation Group	**In/Out**	**OpCode**	**Description**	**Operation Group**	**In/Out**	**OpCode**	**Description**
01	0	000	ESP Outbound	01	1	000	ESP Inbound
01	0	001	AH Outbound	01	1	001	AH Inbound
01	0	010	Reserved for Ipcomp	01	1	010	Reserved for Ipcomp
01	0	011	Reserved for MPPE	01	1	011	Reserved for MPPE
01	0	100	SSL Outbound (See **Note**)	01	1	100	SSL Inbound (See **Note**)
01	0	101	TLS Outbound (See **Note**)	01	1	101	TLS Inbound (See **Note**)
01	0	110	Reserved for WTLS	01	1	110	Reserved for WTLS
01	0	111	SRTP Outbound (See **Note**)	01	1	111	SRTP Inbound (See **Note**)
Note: This is not a full protocol operation. For SSL/TLS and SRTP no Headers processing is performed in hardware.							

17.2.27 SA Command 1 Register (CRYP0_SA_CMD_1)

CRYP0_SA_CMD_1 contains the minor control bits for defining a cryptographic operation.

See the companion register CRYP0_SA_CMD_0.

CRYP0_SA_CMD_1 is Write-only and always return zero on a read by the processor.

Address: 0x0 E011 0604

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-27. SA Command 1 Register (CRYP0_SA_CMD_1)

Bit	Mnemonic	Description	Comments
31	AESCM	AES Counter Mode See description of bits 9:8.	
30	SARC4S	Save ARC4 State Controls whether the PE writes the ARC4 State data back out to the SA Record. 0 Do not save the ARC4 state 1 Save the ARC4 State	This bit is normally set for Stateful ARC4 and not set for Stateless.
29	ARC4SS	ARC4 Stateless/Stateful Controls whether the ARC4 engine is running in the Stateless or Stateful mode. 0 Stateless 1 Stateful	**Stateless:** Each packet is processed with a newly initialized ARC4 key taken from the Key field of the SA record. In this mode, the state information from the SA is never read. **Stateful:** When bit 3, Init ARC4, of the CRYP0_PE_CTLST register is 1, the ARC4 algorithm initializes using the Key specified in the SA record. When the Init ARC4 bit is 0, the ARC4 context is read from the ARC4 State field of the SA record and the i/j Pointer field of the SA. The encryption/decryption processing continues from this previous algorithm state.

Figure 17-27. SA Command 1 Register (CRYP0_SA_CMD_1) (continued)

Bit	Mnemonic	Description	Comments
28:24	ARC4KL	**28:24** ARC4 Key Length Selects the key length, in bytes, for an ARC4 operation. Valid settings range from 1–16 corresponding to 8 bits–128 bits. Note that an ARC4 key longer than 128-bits may be achieved by creating it outside to the SECURITY ENGINE. The external key must be replicated outside the SECURITY ENGINE to fill a 256-byte ARC4 Context and then have the ARC4 key scheduling applied. Finally, the 256-byte table is placed in a State record and the i and j pointers written to the SA (initial i = 1, j = 0). Then use bit#29 to specify 'Stateful' in order to use the key. For SSL and TLS operations specifically key sizes of 5-bytes and 16-bytes are used.	
	AESKL	**26:24** AES Key Length: Selects the size of the key data used for AES operations only. The key length changes in increments of 64 bits: 000 - 001 Reserved 010 128 bits 011 192 bits 100 256 bits 101 – 111 Reserved	
23:16	HCO	Hash/Encryption Offset Specifies the offset, in words, between the hash data and the encryption/decryption data for Basic operations. In the case of Outbound operations, the data to be hashed is assumed to come first, with an offset to the beginning of encrypted data. For Inbound operations, the data to be hashed is assumed to come first, with an offset to the beginning of data to decrypted. For SRTP there is always an offset. For other operations these bits are not used (a default value is applied by the PE). If an IV is loaded through the input buffer, the Hash/Encryption Offset must include the IV. In the case of DES and Triple-DES and AES-CTR operations, the offset is 0x02 words. In the case of AES-CBC and AES-ICM, the offset is 0x04 words.	
15	SAREV	SA Revision Specifies the revision of the SA structure. 0 Revision 0 1 Revision 1	Revision 0 is the legacy revision that is maintained for compatibility with existing code. Revision 1 is the recommended revision, which supports AES as well as DES, and matches the current register layout.
14:13		Reserved	

Figure 17-27. SA Command 1 Register (CRYP0_SA_CMD_1) (continued)

Bit	Mnemonic	Description	Comments
12	HMAC	HMAC/IP Options Muting Control This bit has two functions: **Basic operations that include hashing:** Enables the HMAC processing, which calls for an outer hash operation to occur. 0 Standard hash processing 1 HMAC processing **IPsec AH operations:** Controls Mutable-bit processing on IPv4 options and IPv6 extension headers. 0 Enables Mutable-bit processing on options and extension headers 1 Disables mutable-bit processing	For HMAC operations, pre-computed hash digests must be used and written to the inner and outer digest fields in the SA-Record. If the Hash Final bit in the descriptor is 0 this bit is don't care (overruled) and a standard hash is processed. In case the Hash Final bit in the descriptor is 1 and standard hash processing is selected, the data is hashed and the required hash padding is appended. In case the Hash Final bit in the descriptor is 1 and HMAC is selected, an outer hash operation is applied. Mutable-bit processing for the IP header itself is specified in the MBP field. This bit is don't care (overruled) for ESP, SSL, TLS and SRTP operations. For IPv6 IPsec AH operations, Inbound and Outbound, a Destination Option header after the AH header is never muted.
11:10	CFDBK	Cryptographic Feedback Mode Selects the cryptographic feedback mode for OFB and CFB modes. 00 64-bit OFB/CFB for DES/Triple-DES 01 8-bit CFB for DES/Triple-DES 10 1-bit CFB for DES/Triple-DES 11 128-bit CFB for AES	
9:8	CMODE	Cryptographic Mode Selects the encryption mode for Basic Encryption operations. 000 ECB (Electronic Code Book) 001 CBC (Cipher Block Chaining) 010 OFB (Output Feed Back) 011 CFB (Cipher Feed Back) 100 AES-CTR (32-bit Counter, Counter Mode for IPsec) 101 AES-ICM (16-bit Counter, Integer Counter Mode for SRTP) 110 – 111 Reserved.	
7:6	SRBID	State BusID Selects the Bus ID where the State data will be located.	In the security function, these bits are ignored (fixed to 01). These bits always return 01 on a read by the processor.
5	MBP	IP Header Mutable Bit Handling Controls Mutable-bit zeroing on the IP header as specified in RFC 2402 and RFC 2460, the AH Protocol RFCs to allow the security processor to determine where the AH header is located. 0 Enable Mutable-bit processing 1 Disable Mutable-bit processing	The processor should replace predictable fields before offering the packet to the security function. After the packet is returned, the processor should put the initial values back again. For example, the security processor does not automatically adjust the dest_addr field of the base header to the final destination address according to the routing entries. Similarly, the Routing Header extension (type 0) is also not adjusted. This bit specifies Mutable-bit processing only for the IP header. HMAC in this register controls whether the IPv4 options and IPv6 extension headers are also muted. This bit is ignored for all non-AH operations. For Hash operations that do not require muting, this bit should always be set to 1.

Figure 17-27. SA Command 1 Register (CRYP0_SA_CMD_1) (continued)

Bit	Mnemonic	Description	Comments
4	IP4IP6	IPv4/IPv6 Selects the IP protocol version for this packet flow. 0 IPv4 1 IPv6	This is required for Mutable-bit processing (see MBP above) and in the AH protocol to allow the security function to determine where the AH header is located.
3	CRYPO	Copy Inbound Pad to Output Controls whether the PE transfers padding information from the input buffer to the output buffer. 0 Pad is not copied to the output buffer. 1 Pad is copied to the output buffer.	This bit is only applicable for the following operations and the pad is defined as all data: IPsec ESP Inbound: between the payload and the ICV. SSL Inbound: after the payload (this includes the MAC). TSL Inbound: after the payload (this includes the MAC). Basic Decryption: after the payload. Basic Hash-Decryption: after the payload.
2	CRYPLO	Copy Payload to Output Controls whether the PE transfers payload information from the input buffer to the output buffer. 0 Payload is not copied to the output buffer 1 Payload is copied to the output buffer	This bit is only applicable for the following operations and the payload is: IPsec AH Inbound: defined as all data after the AH. Except in case a second Destination Option Header directly follows the AH it is defined as part of the Header (refer to RFC 2402). IPsec AH Outbound: defined the AH Header and the payload, excluding the second Destination Header. Basic Hash: the entire data to be hashed. For all other operations the payload is always copied from the input buffer to the output buffer.
1	CRYPHO	Copy Header to Output Controls whether the PE transfers header information from the input buffer to the output buffer. 0 Header is not copied to the output buffer 1 Header is copied to the output buffer	This bit is only applicable for the following operations and the header is defined as the: IPsec ESP Inbound: hash/encryption offset data (authenticated only). IPsec AH Outbound: data from the start of a packet to the beginning of the ICV field in the AH header and, if available, includes the Destination Header after the ICV. IPsec AH Inbound: data from the start of a packet to the end of the ICV field and, if available, includes the Destination Header after the ICV. If bits 1 and 2 are both 0, then only the ICV will be returned. SSL Outbound and Inbound: Sequence Number, Type and Length fields. This bit must be set for SSL-Null encryption. TLS Outbound and Inbound: Sequence Number, Type and Length fields. This bit must be set for TLS-Null encryption. SRTP Outbound and Inbound: Bypass Offset, the data from the start of the packet to the start of the payload. In general this bit will be set to 0 since no fields in the header need to be updated. Basic Encryption-Hash and Basic Hash-Decryption: Hash/Cryption Offset data (authenticated only).
0		Reserved	

17.2.28 SA Key x Low/High Registers (CRYP0_SA_KEYx_L/H)

There are four KEY register pairs numbered 1 through 4 (x). These eight registers contain the 64-bit cryptograhic keys. Bit 31:0 are in the low (L) register while bits 63:32 are in the high (H) register.

Address: 0x0 E011 0610 – CRYP0_SA_KEY1_L
0x0 E011 0614 – CRYP0_SA_KEY1_H
0x0 E011 0618 – CRYP0_SA_KEY2_L
0x0 E011 061C – CRYP0_SA_KEY2_H
0x0 E011 0620 – CRYP0_SA_KEY3_L
0x0 E011 0624 – CRYP0_SA_KEY3_H
0x0 E011 0628 – CRYP0_SA_KEY4_L
0x0 E011 062C – CRYP0_SA_KEY4_H

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-28. SA Key x Low/High Registers (CRYP0_SA_KEYx_L/H)

Bit	Mnemonic	Description	Comments
31:0		DES/Triple-DES/ARC4/AES Key Key bits 63:32 are in CRYP0_SA_KEYx_H. Key bits 31:0 are in CRYP0_SA_KEYx_L.	For a Single-DES key, only KEY1 is used. For a Triple-DES key, the first three pairs of registers (KEY1, KEY2 and KEY3) are used. For ARC4 the first two pair of registers (KEY1 and KEY2) are used. For an AES, a: a. 128-bit key uses KEY1 and KEY2 b. 192-bit key uses KEY1, KEY2, and KEY3 c. 256-bit key uses KEY1, KEY2, KEY3, and KEY4.

17.2.29 SA Inner Hash Digest x Registers (CRYP0_SA_IH_Dx)

These five registers (x = 0–4) contain the 160-bit inner hash digest value.

See also *SA Inner Hash x (mirror of CRYP0_SA_IH_Dx) Registers (CRYP0_SA_IH_x)* on page 354

Address: 0x0 E011 0630 – CRYP0_SA_IH_D0
0x0 E011 0634 – CRYP0_SA_IH_D1
0x0 E011 0638 – CRYP0_SA_IH_D2
0x0 E011 063C – CRYP0_SA_IH_D3
0x0 E011 0640 – CRYP0_SA_IH_D4

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-29. SA Inner Hash Digest x Registers (CRYP0_SA_IH_Dx)

Bit	Mnemonic	Description	Comments
31:0		Inner Hash Digest Hash bits 31:0 are in CRYP0_SA_IH_D0. Hash bits 63:32 are in CRYP0_SA_IH_D1. Hash bits 95:64 are in CRYP0_SA_IH_D2. Hash bits 127:96 are in CRYP0_SA_IH_D3. Hash bits 159:128 are in CRYP0_SA_IH_D4.	For an MD5 hash, only the first four registers are used. For a SHA-1 hash, all five registers are used. These are used both for entering a starting hash state, as well as for reading the interim or final hash digest. For IPsec operations with authentication or basic hash or HMAC with Load Hash State = 00 (from the SA), the pre-computed inner hash digest is written here. When a hash pre-compute is written here, the starting hash byte count is automatically set to 64 (0x40) to indicate that 64 bytes have been processed through the hash.

17.2.30 SA Inner Outer Digest x Registers (CRYP0_SA_OH_Dx)

When *writing*, these five registers (x = 0–4) contain the 160-bit pre-computed outer hash digest for IPsec operations with authentication, or basic HMAC operations with Load Hash from SA set. They are used for writing the starting or interim outer hash digest, and for HMAC processing only.

When *reading*, these five registers contain the MAC hash result used for SSL and TLS and has the same functionality as the ICV register for IPsec.

Address: 0x0 E011 0644 – CRYP0_SA_OH_D0
0x0 E011 0648 – CRYP0_SA_OH_D1
0x0 E011 064C – CRYP0_SA_OH_D2
0x0 E011 0650 – CRYP0_SA_OH_D3
0x0 E011 0654 – CRYP0_SA_OH_D4

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-30. SA Inner Hash Digest x Registers (CRYP0_SA_OH_Dx)

Bit	Mnemonic	Description	Comments
31:0		Outer Hash Digest/MAC Hash Result Bits 31:0 are in CRYP0_SA_OH_D0. Bits 63:32 are in CRYP0_SA_OH_D1. Bits 95:64 are in CRYP0_SA_OH_D2. Bits 127:96 are in CRYP0_SA_OH_D3. Bits 159:128 are in CRYP0_SA_OH_D4.	When *writing*; For MD5, only the first four registers are used. For SHA-1, all five registers are used. When *reading*: For MD5 the MAC is 16 bytes. For SHA the MAC is 20 bytes.

17.2.31 SA IPsec SPI Register (CRYP0_SA_SPI)

This register has two functions: For IPsec operations, it is written with the SPI (Security Parameters Index) associated with the inbound or outbound flow. In addition, it is the trigger register for indicating that the key and hash digests have been written.

This register *must* be written for all Direct Host DMA packet operations, even if the SPI is not used (any value may be written for non-IPSEC operations). This register should be written "in sequence" as the entire SA record is written. Therefore, it must be written after the key and hash digests, but it can be written before the Sequence Number and Mask registers.

Address: 0x0 E011 0658

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-31. SA IPsec SPI Register (CRYP0_SA_SPI)

Bit	Mnemonic	Description	Comments
31:0		SA IPsec SPI Used for IPsec ESP and AH operations to specify the Security Parameters Index (SPI) value that is to be placed in the ESP or AH header.	The description applies only to the Direct Host DMA mode. For Autonomous Ring mode and Target Command mode, the security function extracts the SPI from of the SA-Record. There is no need to read back this value at the end of an operation, since the security function will not change it.

17.2.32 SA IPsec Sequence Number Register (CRYP0_SA_SEQ)

This register is used for IPsec ESP and AH operations to specify the anti-replay sequence number value that is to be placed in the ESP or AH header for outbound operations, or to be checked against for inbound packets. The security function manages this counter value for both inbound and outbound operations.

Address: 0x0 E011 065C

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-32. SA IPsec Sequence Number Register (CRYP0_SA_SEQ)

Bit	Mnemonic	Description	Comments
31:0		Sequence Number **Outbound:** The processor writes the counter value stored in the SA record into this register when starting an IPsec operation. The security function automatically increments the count. Upon completion, the processor can read this value and write it to the SA record. **Inbound:** The processor writes the counter value stored in the SA record into this register when starting an IPsec operation. The security function automatically performs the specified inbound processing (per RFC2402 and RFC2406) as it processes the packet. As a result, the expected count value might or might not be updated during processing. Upon completion, the processor reads back this value and writes it to the SA record.	The description only applies to the Direct Host DMA mode. For Autonomous Ring mode and Target Command mode, the sequence number is updated by the PE.

17.2.33 SA IPsec Sequence Number Mask High/Low Registers (CRYP0_SA_SEQMKH/L)

This pair of registers is used for IPsec ESP and AH operations to specify the anti-replay sequence number mask value for inbound operations. The security function manages this counter value automatically.

Address: 0x0 E011 0660 and 0x0 E011 0664

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-33. SA IPsec Sequence Number Mask High and Low Registers (CRYP0_SA_SEQMKH/L

Bit	Mnemonic	Description	Comments
31:0		Sequence Number Mask Bits 31:0 are in CRYP0_SA_SEQMKL Bits 63:32 are in CRYP0_SA_SEQMKH **Outbound:** Not used. **Inbound:** The processor writes the counter value stored in the SA record into this register upon starting an IPsec operation. The security function automatically performs the specified inbound processing (per RFC2402 and RFC2406) as it processes the packet. As a result, the new mask value might or might not be updated during processing. Upon completion, the processor reads this value and writes it to the SA record.	The description only applies to the Direct Host DMA mode. For Autonomous Ring mode and Target Command mode, the security function extracts the Sequence Number Mask from of the SA-Record.

17.2.34 SA Nonce Value Register (CRYP0_SA_NONCE)

This register is used for AES CTR operations that make use of the Nonce value that is loaded from the SA-record.

Address: 0x0 E011 0668

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-34. SA Nonce Value Register (CRYP0_SA_NONCE)

Bit	Mnemonic	Description	Comments
31:0		Nonce Value	Direct Host DMA mode only.

17.2.35 SA Pointer Register (CRYP0_SA_PNTR)

This register contains the address pointer to the state record.

Address: 0x0 E011 066C

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-35. SA Pointer Register (CRYP0_SA_PNTR)

Bit	Mnemonic	Description	Comments
31:0		Pointer	Direct Host DMA mode only.

17.2.36 SA ARC4 i and j Pointer Register (CRYP0_SA_ARC4IJ)

When starting a new ARC4 operation this register contains the initialization value (0). After processing the ARC4 algorithm, it contains the latest status of the ARC4_I_J_PNTR.

Address: 0x0 E011 0670

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-36. SA ARC4 i and J Pointer Register (CRYP0_SA_ARC4IJ)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15:8	JPNT	j Pointer Pointer into s-box array for swapping bytes with i pointer.	The description applies only to the Direct Host DMA mode. For Autonomous Ring mode and Target Command mode the security function extracts the pointer from of the SA-Record.
7:0	IPNT	i Pointer Pointer into s-box array for swapping bytes with j pointer	The description applies only to the Direct Host DMA mode. For Autonomous Ring mode and Target Command mode the security function extracts the pointer from of the SA-Record.

17.2.37 SA ARC4 State Address Pointer Register (CRYP0_SA_ARC4SB)

This register contains the pointer to externally located ARC4 SBOX data.

Address: 0x0 E011 0674

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-37. SA ARC4 State Address Pointer Register (CRYP0_SA_ARC4SB)

Bit	Mnemonic	Description	Comments
31:0		ARC4 SBOX Pointer Register	Direct Host DMA mode only.

17.2.38 SA Initialization Vector Registers (CRYP0_SA_IV_x)

These four registers contain the 128-bit initialization vector.

Address: 0x0 E011 06C0 – CRYP0_SA_IV_0
0x0 E011 06C4 – CRYP0_SA_IV_1
0x0 E011 06C8 – CRYP0_SA_IV_2
0x0 E011 06CC – CRYP0_SA_IV_3

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-38. SA Initialization Vector Registers (CRYP0_SA_IV_x)

Bit	Mnemonic	Description	Comments
31:0		Initilization Vector (IV) Bits 31:0 are in CRYP0_SA_IV_0 Bits 63:32 are in CRYP0_SA_IV_1 Bits 95:64 are in CRYP0_SA_IV_2 Bits 127:96 are in CRYP0_SA_IV_3 These registers are used both for entering a starting IV state as well as for reading the interim or final IV. For operations that use cryptographic mode AES-ICM, CRYP0_SA_CMD_1[31, 9:8] is 100, CRYP0_SA_IV_3[31:0] are the block counter. For operations that use crypto mode AES-CTR, CRYP0_SA_CMD_1[31,9:8] is 101, CRYP0_SA_IV_3[31:0] are the block counter and CRYP0_SA_IV_0[31:0] are the Nonce.	Direct Host DMA mode only. For IPsec *outbound* operations, it is recommended that the automatic IV insertion mode be used, which means that these registers are not needed. For IPsec *inbound* operations, the IV is extracted from the header of the packet.

17.2.39 SA Hash Byte Count Register (CRYP0_SA_HASH_B)

This register is used both for entering a starting hash byte count, as well as for reading the interim or final byte count.

For some hash operations, this register is ignored and the byte count is internally set to 64 to indicate that the first 64-byte hash block has been processed using a pre-computed hash state. These operations include:

- All IPsec OpCodes that use authentication (pre-computed inner and outer hash digests are loaded from SA words 10–19)
- Basic Operations with Load Hash from SA specified. For Basic Hash with no HMAC, a pre-computed digest is loaded from SA words 10–14. For Basic HMAC, the inner and outer digests are loaded from SA words 10–19.

Note: IPsec operations can not be suspended in mid-packet and resumed later, and so do not use this register.

Address: 0x0 E011 01D0

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-39. SA Hash Byte Count Register (CRYP0_SA_HASH_B)

Bit	Mnemonic	Description	Comments
31:0		Hash Byte Count	Direct Host DMA mode only.

17.2.40 SA Inner Hash x (mirror of CRYP0_SA_IH_Dx) Registers (CRYP0_SA_IH_x)

See *SA Inner Hash Digest x Registers (CRYP0_SA_IH_Dx)* on page 347.

Address: 0x0 E011 06D4 – CRYP0_SA_IH_0
0x0 E011 06D8 – CRYP0_SA_IH_1
0x0 E011 06DC – CRYP0_SA_IH_2
0x0 E011 06E0 – CRYP0_SA_IH_3
0x0 E011 06E4 – CRYP0_SA_IH_4

Size (bits): 32

Access: R (master and slave)/W (master only)

Reset: 0x0000 0000

Figure 17-40. SA Inner Hash x (mirror of CRYP0_SA_IH_Dx) Registers (CRYP0_SA_IH_x)

Bit	Mnemonic	Description	Comments
31:0		Mirror of CRYP0_SA_IH_Dx	

17.2.41 SA ICV x—HMAC Result Registers (CRYP0_SA_ICV_x)

These registers are used to store the ICV value for IPsec, SSL and TSL operations. For outbound operations they contain the hash result. For inbound operations they contain the ICV that will be checked against the hash result.

For ESP and AH inbound operations, the ICV registers are typically not accessed by software. The ICV value is normally calculated internally and automatically compared against the ICV contained in the packet. The authentication result is then provided in the result descriptor.

For ESP outbound operations, the ICV registers are typically not used, as the ICV value is inserted in the data output buffer directly after the payload and is read out of the data output FIFO along with the processed packet.

For an AH outbound operation, the ICV registers are typically not used, as the ICV value is inserted in the data output buffer directly in front of the payload and is read out of the data output FIFO along with the processed packet. When the packet size is larger than the buffer size, the processed packet is read from the data output buffer and then the ICV is updated in memory.

Address: 0x0 E011 06E8 – CRYP0_SA_ICV_0
0x0 E011 06EC – CRYP0_SA_ICV_1
0x0 E011 06F0 – CRYP0_SA_ICV_2
0x0 E011 06F4 – CRYP0_SA_ICV_3
0x0 E011 06F8 – CRYP0_SA_ICV_4

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-41. SA ICV x—HMAC Result Registers (CRYP0_SA_ICV_x)

Bit	Mnemonic	Description	Comments
31:0		HMAC Result Bits 31:0 are in CRYP0_SA_ICV_0 Bits 63:32 are in CRYP0_SA_ICV_1 Bits 95:64 are in CRYP0_SA_ICV_2 Bits 127:96 are in CRYP0_SA_ICV_3 Bits 159:128 are in CRYP0_SA_ICV_4 Contain the 160-bit ICV for IPsec authentication operations. This is the computed ICV value (as opposed to the ICV extracted from an inbound packet). The ICV contained here differs from the hash digest registers above in that for a SHA-1 hash, the ICV field is endian-swapped from the hash digest field. This is consistent with the IPsec packet format standards.	Direct Host DMA mode only.

17.2.42 TRNG Output Register (CRYP0_TRNG_DATA)

This register contains a word of random data. The processor is responsible for monitoring the True Random Number Generator (TRNG) status register to make sure that the data presented is ready.

Address: 0x0 E012 0100

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-42. TRNG Output Register (CRYP0_TRNG_DATA)

Bit	Mnemonic	Description	Comments
31:0		TRNG Data When CRYP0_TRNG_CTRL[12:11] = 00 the register contains the 32-bit Non-Linear Mixer true random number output. When CRYP0_TRNG_CTRL[12] = 1 the register contains the 64-bit FIPS 140-2 Post-Processor random number output. The first read provides the lower 32 bits, the second read provides the higher 32 bits. When this register is read, the TRNG sets its Busy status bit and automatically begins generating a new random word. When CRYP0_TRNG_CTRL[12:11] = 11 the 64-bit FIPS 140-2 Post-Processor data input must be provided in this register. The first write must provide the lower 32 bits, the second write must provide the higher 32 bits.	See *TRNG Control Register (CRYP0_TRNG_CTRL)* on page 357.

17.2.43 TRNG Status Register (CRYP0_TRNG_STAT)

This register contains the TRNG busy indicator.

Address: 0x0 E012 0104

Size (bits): 32

Access: Read only

Reset: 0x0000 000

Figure 17-43. TRNG Status Register (CRYP0_TRNG_STAT)

Bit	Mnemonic	Description	Comments
31:1		Reserved	
0	B	Busy 0 Valid random number is available in the output register 1 TRNG is busy generating the next random number	The bit is set to 1 when the TRNG Output register is read, and the TRNG starts generating the next random number.

17.2.44 TRNG Control Register (CRYP0_TRNG_CTRL)

This register is used to test the TRNG operation.

Address: 0x0 E012 0108

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-44. TRNG Control Register (CRYP0_TRNG_CTRL)

Bit	Mnemonic	Description	Comments
31:13		Reserved	
12	EPP	Control the Post Processor 0 Disable the Post Processor 1 Enable the Post Processor	See *TRNG Output Register (CRYP0_TRNG_DATA)* on page 356.
11	BTRNG	Bypass TRNG 0 Normal run state 1 Enable direct access to the FIPS 140-2 Post Processor	See *TRNG Output Register (CRYP0_TRNG_DATA)* on page 356.
10	NRLFSR	nReset the Linear Feedback Shift Registers (LFSRs) 0 Reset the LFSRs within the TRNG 1 Normal run state	
9	TLFSR	Test LFSRs 0 Normal run state 1 Initiates a self-test of the LFSRs in the TRNG where the system clock replaces the ring oscillators as the LFSR source	
8	TALM	Test Alarm 0 Normal run state 1 Forces a simulated alarm state	The alarm counter increments on each system clock.
7	SC	Short Cycle 0 Normal run state 1 Shortens the state timers for simulation and testing	
6	TCNT	Clock on 0 Normal run state 1 Enables the test clock to be the system clock and allows the ring outputs to be processed in test mode	
5	DALM	Disable Alarm 0 Normal run state 1 Disables TRNG alarm reporting and does not reset the TRNG when the alarm is triggered	
4	TRNG2	Ring Oscillator #2 test 0 Disable test 1 Enable test	
3	TRNG1	Ring Oscillator #1 test 0 Disable test 1 Enable test	

Figure 17-44. TRNG Control Register (CRYP0_TRNG_CTRL) (continued)

Bit	Mnemonic	Description	Comments
2	TRUN	Enable Test Run 0 Normal run state 1 Starts the test state machine sequence	All cycles of the TRNG Finite State Machine (FSM) are tested.
1	TMODE	Enable Test Mode 0 Normal run state 1 Enables the internal TRNG registers to be written or read by the processor	Written registers: Entropy, Seed and Counter Read registers: Entropy, Seed, Counter, and LFSR
0	TRNG	Enable Test Ring Output 0 Normal run state 1 Ring oscillator output is not passed through the LFSRs in generating the random number	

17.2.45 TRNG Entropy A Register (CRYP0_TRNG_ENTA)

This register provides the read access to the Entropy A register for test and diagnostic purposes. It is accessible in Test Mode only.

Address: 0x0 E012 010C

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-45. TRNG Entropy A Register (CRYP0_TRNG_ENTA)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15:0	EA	Entropy A Contains the entropy output of the Ring Oscillator/ LFSR1 pair.	

17.2.46 TRNG Entropy B Register (CRYP0_TRNG_ENTB)

This register provides the read access to the Entropy B register for test and diagnostic purposes. It is accessible by the processor in Test Mode only.

Address: 0x0 E012 0110

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-46. TRNG Entropy B Register (CRYP0_TRNG_ENTB)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15:0	EB	Entropy B Contains the entropy output of the Ring Oscillator/ LFSR2 pair.	

17.2.47 TRNG Test Seed x Registers (CRYP0_TRNG_Xx)

These registers provide read access to the X (x = 0,1, or 2) registers for test and diagnostic purposes. They are accessible in Test Mode only.

Address: 0x0 E012 0114 – CRYP0_TRNG_X0
0x0 E012 0118 – CRYP0_TRNG_X1
0x0 E012 011C – CRYP0_TRNG_X2

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-47. TRNG Test Seed x Registers (CRYP0_TRNG_Xx)

Bit	Mnemonic	Description	Comments
31:0	RNGX	Xx Registers X0 = LFSR[31:0] X1 = LFSR[63:32]: X2 = LFSR[80:64]	X0 – 32 least significant state bits of the X LFSR. X1 – 32 middle state bits of the X LFSR. X2 – 17 most significant state bits of the X LFSR.

17.2.48 TRNG Counter Register (CRYP0_TRNG_CNTR)

This register provides the access to the Counter register for test and diagnostic purposes. It is accessible in Test Mode only.

Address: 0x0 E012 0120

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-48. TRNG Counter Register (CRYP0_TRNG_CNTR)

Bit	Mnemonic	Description	Comments
31:24		Reserved	
23:0	CNT	Counter State machine counter for test and diagnostic purposes.	

17.2.49 TRNG Alarm Counter Register (CRYP0_TRNG_ALRM)

This register accumulates the number of comparison faults detected on the output data. An output data block that is identical to the prior output generates a comparison fault. A very small number of faults are to be expected statistically. The function reading this register must apply an algorithm to determine what is significant. The count value is reset to zero whenever an output sample does not generate a comparison fault.

Address: 0x0 E012 0124

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-49. TRNG Alarm Counter Register (CRYP0_TRNG_ALRM)

Bit	Mnemonic	Description	Comments
31:8		Reserved	
7:0	ALMCNT	Alarm Count Provides a non-cumulative count of the number of comparison faults that have been detected on the TRNG output.	

17.2.50 TRNG Configuration Register (CRYP0_TRNG_CFG)

This register configures the TRNG.

Address: 0x0 E012 0128

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-50. TRNG Configuration Register (CRYP0_TRNG_CFG)

Bit	Mnemonic	Description	Comments
31:12		Reserved	
11:6	RSTCNT	Reset Count Threshold of consecutive unchanging output bits for resetting LFSRs.	The counter value is set to 0x01 to avoid having the TRNG reset immediately after start-up.
5:3	R2DLY	Ring 2 Delay Selects delay for Ring Oscillator 2	
2:0	R1DLY	Ring 1 Delay Selects delay for Ring Oscillator 1	

17.2.51 TRNG Test Read of LFSR 0 Low/High Registers (CRYP0_TRNG_LF0L/H)

These registers provide access to Linear Feedback Shift Register (LFSR) 0 for test and diagnostic purposes. They are accessible in Test Mode only.

Address: 0x0 E012 012C – CRYP0_TRNG_LF0L
0x0 E012 0130 – CRYP0_TRNG_LF0H

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-51. TRNG Test Read of LFSR 0 Low Register (CRYP0_TRNG_LF0L)

Bit	Mnemonic	Description	Comments
31:0	LFSR0	32 least significant state bits of LFSR 0 [31:0]	

Figure 17-52. TRNG Test Read of LFSR 0 High Register (CRYP0_TRNG_LF0H)

Bit	Mnemonic	Description	Comments
31:17		Reserved	
16:0	LFSR0	17 most significant state bits of LFSR 0 [48:32]	

17.2.52 TRNG Test Read of LFSR 1 Low/High Registers (CRYP0_TRNG_LF1L/H)

These registers provide access to LFSR 1 for test and diagnostic purposes. They are accessible in Test Mode only.

Address: 0x0 E012 0134 – CRYP0_TRNG_LF1L
0x0 E012 0138 – CRYP0_TRNG_LF1H

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-53. TRNG Test Read of LFSR 1 Low Register (CRYP0_TRNG_LF1L)

Bit	Mnemonic	Description	Comments
31:0	LFSR1	32 least significant state bits of LFSR 1 [31:0]	

Figure 17-54. TRNG Test Read of LFSR 1 High Register (CRYP0_TRNG_LF1H)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15:0	LFSR1	16 most significant state bits of LFSR 1 [47:32]	

17.2.53 TRNG Triple DES Key 0 Low/High Registers (CRYP0_TRNG_K0_L/H)

These registers provides the Triple-DES Key 0 to the post-processor. The key register is implemented as a 64-bit maximum length LSFR to generate pseudo-random key values and is updated at the end of a TRNG operation.

Address: 0x0 E012 013C – CRYP0_TRNG_K0_L
0x0 E012 0140 – CRYP0_TRNG_K0_H

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-55. TRNG Triple DES Key 0 Low/High Registers (CRYP0_TRNG_K0_L/H)

Bit	Mnemonic	Description	Comments
31:0		Key 0 CRYP0_TRNG_K0_L – bits 31:0 (lsb) CRYP0_TRNG_K0_H – bits 63:32 (msb)	If the processor reads this register it returns an undefined value.

17.2.54 TRNG Triple DES Key 1 Low/High Registers (CRYP0_TRNG_K1_L/H)

These registers provides the Triple-DES Key 1 to the post-processor. The key register is implemented as a 64-bit maximum length LSFR to generate pseudo-random key values and is updated at the end of a TRNG operation.

Address: 0x0 E012 0144 – CRYP0_TRNG_K1_L
0x0 E012 0148 – CRYP0_TRNG_K1_H

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-56. TRNG Triple DES Key 1 Low/High Registers (CRYP0_TRNG_K1_L/H)

Bit	Mnemonic	Description	Comments
31:0		Key 1 CRYP0_TRNG_K1_L – bits 31:0 (lsb) CRYP0_TRNG_K1_H – bits 63:32 (msb)	If the processor reads this register it returns an undefined value.

17.2.55 TRNG Initialization Vector Low/High Registers (CRYP0_TRNG_IV_L/H)

These registers provides an initialization vector (IV) to the post-processor. For FIPS 140-2 compliance, the DES post-processor must operate in CBC (cipher-block-chaining) mode.

Address: 0x0 E012 014C – CRYP0_TRNG_IV_L
0x0 E012 0150 – CRYP0_TRNG_IV_H

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-57. TRNG Initialization Vector Low/High Registers (CRYP0_TRNG_IV_L/H)

Bit	Mnemonic	Description	Comments
31:0		Key 1 CRYP0_TRNG_IV_L – bits 31:0 (lsb) CRYP0_TRNG_IV_H – bits 63:32 (msb)	If the processor reads this register it returns an undefined value.

17.2.56 PKA x Vector Address Register (CRYP0_PKA_x_PTR)

These registers provide the address of the x vector (x= A, B, C, or D) in the PKM.

Address: 0x0 E013 0800 – CRYP0_PKA_A_PTR
0x0 E013 0804 – CRYP0_PKA_B_PTR
0x0 E013 0808 – CRYP0_PKA_C_PTR
0x0 E013 080C – CRYP0_PKA_D_PTR

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-58. PKA x Vector Address Register (CRYP0_PKA_x_PTR)

Bit	Mnemonic	Description	Comments
31:11		Reserved	
10:0	xADDR (x = A, B, C, or D)	x Address Address of the least significant word of the x vector.	

17.2.57 PKA x Vector Length Register (CRYP0_PKA_x_LEN)

These registers provide the length of the x vector (x= A or B) in the PKM.

Address: 0x0 E013 0810 – CRYP0_PKA_A_LEN
0x0 E013 0814 – CRYP0_PKA_B_LEN

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-59. PKA x Vector Length Register (CRYP0_PKA_x_LEN)

Bit	Mnemonic	Description	Comments
31:9		Reserved	
8:0	xL (x = A or B)	x Length Length of the x vector in 32-bit words.	Valid settings range from 0 – 512.

17.2.58 PKA Shift Register (CRYP0_PKA_SHIFT)

This register provides the operation code for the public key operation.

Address: 0x0 E013 0818

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-60. PKA Shift Register (CRYP0_PKA_SHIFT)

Bit	Mnemonic	Description	Comments
31:5		Reserved	
4:0	SV	Shift Value Specifies the number of bits to shift in the arithmetic operation	

17.2.59 PKA Function Code Register (CRYP0_PKA_FUNC)

This register provides the function code for the public key operation.

Address: 0x0 E013 081C

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-61. PKA Function Code Register (CRYP0_PKA_FUNC)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15	RUN	Run 0 Operation is complete 1 Start the requested operation (bits 14:0)	
14	E2ACT	Exponentiate (2-bit ACT)	
13	E4ACT	Exponentiate (4-bit ACT)	
12	RED	Reduce	Use for test purposes only.
11	CPY	Copy	
10	CPR	Compare	
9	MOD	Mod	Use for test purposes only.
8	DIV	Divide	
7	LSHFT	Left Shift	
6	RSHFT	Right Shift	
5	SUB	Subtract	
4	ADD	Add	
3	DI	Dusse Inverse	Use for test purposes only.
2	SQR	Square	
1	DMLT	Dusse Multiply	Use for test purposes only.
0	MLT	Multiply	

Figures 17-62 and Figures 17-63 define the mathematical definitions for the functions within the PKA Function Code Register.

Figure 17-62. Mathematical Operations

Function	Operation	Vector A	Vector B	Vector C	Vector D
Multiply	A x B --> C	Multiplicand	Multiplier	Product	N/A
Square	A * A --> C	Multiplicand	N/A	Result	N/A
Add	A + B --> C	Addend	Addend	Sum	N/A
Subtract	A – B --> C	Minuend	Subtrahend	Difference	N/A
Right Shift	A >> Shift --> D	Input	N/A	Result	N/A
Left Shift	A << Shift --> D	Input	N/A	Result	N/A
Divide	A mod B --> C A div B --> D	Dividend	Divisor	Remainder	Quotient
Modulo	A mod B --> C	Dividend	Divisor	Remainder	N/A
Compare	A = B A < B A > B	Input 1	Input 2	N/A	N/A
Copy	A --> C	Input	N/A	Result	N/A
Exp (2-bit ACT)	D = C^ mod B	Input 1	Input 2	Input 3	Result
Exp (4-bit ACT)	D = C^ mod B	Input 1	Input 2	Input 3	Result

Figure 17-63. Operations Description

Definitions
PKA_A_PTR base address of A (0 <= PKA_A_PTR < 2^ 10)
PKA_B_PTR base address of B (0 <= PKA_B_PTR < 2^ 10)
PKA_C_PTR base address of C (0 <= PKA_C_PTR < 2^ 10)
PKA_A_LENGTH length of A in 32-bit words (0 < PKA_A_LENGTH <= 255)
PKA_B_LENGTH length of B in 32-bit words (0 < PKA_B_LENGTH <= 255)
PKA_DIV_MSW address of result
PKA_SHIFT number of bits to shift to input A
Operational Description
Multiply and Divide: Result length is (PKA_A_LENGTH + PKA_B_LENGTH + 8) in 32-bit words
Multiply: A and C may not overlap B and C may not overlap

Figure 17-63. Operations Description (continued)

Square:
1 < PKA_A_LENGTH <= 255
Result length is PKA_A_LENGTH + PKA_B_LENGTH + 8
A and C may not overlap unless they start at the same address

Add:
if PKA_A_LENGTH >= PKA_B_LENGTH then
(0<PKA_A_ LENGTH, PKA_B_LENGTH <= 256)
Result length is PKA_A_LENGTH + 1 in 32-bit words
A and C may not overlap unless they start at the same address
B and C may not overlap unless they start at the same address

Subtract:
PKA_A_LENGTH >= PKA_B_LENGTH
Result length is Max (PKA_A_LENGTH) in 32-bit words
A and C may not overlap unless they start at the same address
B and C may not overlap unless they start at the same address

Shift:
A > 0
0 <= PKA_SHIFT < 32
Result length is PKA_A_LENGTH in 32-bit words
A and C may not overlap unless they start at the same address

Divide:
B /= 0, A >= B
1 < PKA_A_LENGTH <= 255
1 < PKA_B_LENGTH <= 255
PKA_MOD_MSW address of most significant, non-zero word of remainder
PKA_DIV_MSW address of most significant, non-zero word of quotient
Remainder length (PKA_A_LENGTH - PKA_B_LENGTH + 1) in 32-bit words
Quotient length (PKA_A_LENGTH + 1) in 32-bit words

Compare:
C[0] <= A = B, C[1] <=A, C[2] <= A > B
Result: bit [0] <= A = B, bit[1] <= A < B, bit [2] <= A > B

Copy:
0 < PKA_A_LENGTH <= 256
Result length is PKA_A_LENGTH in 32-bit words

Exponentiation (2-bit ACT):
A /= 0, C /= 0, B[0] = 1, B > C
0 < PKA_A_LENGTH <= 128
1 < PKA_A_LENGTH <= 128
PKA_A_LENGTH < PKA_B_LENGTH
Result length ((20 x PKA_B_LENGTH) + 48) in 32-bit words
B cannot be 0, must be odd and most significant word of B cannot be zero
Exponent A, modulus B, base C and result D may not overlap

Exponentiation (4-bit ACT):
A /= 0, C /= 0, B[0] = 1, B > C
0 < PKA_A_LENGTH <= 128
1 < PKA_A_LENGTH <= 128
PKA_A_LENGTH < PKA_B_LENGTH
Result length ((8 x PKA_B_LENGTH) + 48) in 32-bit words
B cannot be 0, must be odd and most significant word of B cannot be zero
Exponent A, modulus B, base C and result D may not overlap

17.2.60 PKA Comparison Result Register (CRYP0_PKA_COMP)

This register provides the result of Compare operations.

Address: 0x0 E013 0820

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-64. PKA Comparison Result Register (CRYP0_PKA_COMP)

Bit	Mnemonic	Description	Comments
31:3		Reserved	
2	SUP	A > B 0 A not > B 1 A > B	
1	INF	A < B 0 A not < B 1 A < B	
0	EQL	A = B 0 A ≠ B 1 A = B	

17.2.61 PKA Quotient MSW Register (CRYP0_PKA_DIV)

This register returns the PKM address of the most significant, non-zero 32 bit word of the quotient vector. For an all-zero result vector, bit 11 is asserted, and the MSW address should be ignored. This register is only applicable to divide operations.

The divide operation presents a special case because two result vectors are generated—the quotient and remainder (modulus). This register corresponds to the quotient vector. The MSW of the remainder vector is specified in the CRYP0_PKA_MOD register.

.**Address:** 0x0 E013 0824

Size (bits): 32

Access: Read only

Reset: 0x0000 0800

Figure 17-65. PKA Quotient MSW Register (CRYP0_PKA_DIV)

Bit	Mnemonic	Description	Comments
32:12		Reserved	
11	RZERO	Result Zero 0 Non-zero quotient vector 1 Zero quotient vector	
10:0	DIVMSW	DIV MSW Address PKM address of the most significant, non-zero 32-bit word of the quotient vector.	

17.2.62 PKA Remainder MSW Register (CRYP0_PKA_MOD)

This register returns the PKM address of the most significant, non-zero 32 bit word of the remainder vector. For an all-zero remainder vector, bit 11 is asserted, and the MSW address should be ignored. This register is only applicable to divide operations.

The divide operation presents a special case because two result vectors are generated—the quotient and remainder (modulus). This register corresponds to the remainder vector. The MSW of the quotient vector is specified in the CRYP0_PKA_DIV register.

Address: 0x0 E013 0828

Size (bits): 32

Access: Read only

Reset: 0x0000 0800

Figure 17-66. PKA Remainder MSW Register (CRYP0_PKA_MOD)

Bit	Mnemonic	Description	Comments
32:12		Reserved	
11	RZERO	Result Zero 0 Non-zero result vector 1 Zero result vector	
10:0	RMSWA	Result MSW Address PKM address of the most significant, non-zero 32-bit word of the result vector.	

Example:

The following is an example of how to use the PKA multiply function:

The security function is little-endian, so that is how the data must be applied and the result interpreted.

The A operand data = 0x0000 0003 0000 0001 (where 01 is LSB)
The B operand data = 0x0000 0004 0000 0002 (where 02 is LSB)

PKA_A_PTR = 0x390
PKA_B_PTR = 0x27E
PKA_C_PTR = 0x0BA

PKA_A_LENGTH = 0x2
PKA_B_LENGTH = 0x2

PKA Memory

0x390* = 0x00000001
0x391 = 0x00000003

0x27E = 0x00000002
0x27F = 0x00000004

0x0BA = 0x00000002
0x0BB = 0x0000000A
0x0BC = 0x0000000C
0x0BD = 0x00000000

(*word addresses)
So the final C operand data = 0x0000 0000 0000 000C 0000 000A 0000 0002
(where 02 is LSB)

Compare with the paper calculation method:

```
                    0x0000000300000001
                    0x0000000400000002
                    ---------------------------
                    0x0000000600000002
          + 0x0000000C0000000400000000
-----------------------------------------------------
0x000000000000000C0000000A00000002
```

17.2.63 Interrupt Unmasked and Masked Status Registers (CRYP0_INT_UNMSK/MSK)

These registers provide interrupt status visibility to the processor. CRYP0_INT_UNMSK applies *prior* to the interrupt mask being applied. CRYP0_INT_MSK applies *after* to the interrupt mask is applied. Using CRYP0_INT_UNMSK, the processor can view all potential sources of incoming interrupts. Using CRYP0_INT_MSK, the processor can view all interrupts directed to the UIC. All of these sources, whether masked in or out, will be latched in these registers and must be cleared using the CRYP0_INT_MSK register in order to capture a subsequent event. A bit set to 1 indicates that the associated interrupt is present.

Address: 0x0 E015 00A0 – CRYP0_INT_UNMSK
0x0 E015 00A4 – CRYP0_INT_MSK

Size (bits): 32

Access: Read only – CRYP0_INT_UNMSK
R/W – CRYP0_INT_MSK (Reading views the interrupts. Writing clears the interrupts. Writing 1 to a bit clears the corresponding interrupt. Writing 0 leaves the interrupt latch unchanged.)

Reset: 0x0000 0000

Figure 17-67. Interrupt Unmasked and Masked Status Registers (CRYP0_INT_UNMSK/MSK)

Bit	Mnemonic	Description	Comments
32:16		Reserved	
15	DESC	Command to the Packet Engine to fetch the next Packet Descriptor from the PDR.	Applicable only when the Packet Fetch Engine has been enabled with a non-0 CRYP0_PE_RING_S register setting. It is not an internally generated interrupt. It is used for test purposes only to verify that the DESC command in SDR0_CRYP0 has been received correctly by the packet engine.
14	DESCRD	Specified number of Descriptors (1 – 63) have completed processing. Also occurs if a Descriptor has just been processed and there are no more input Descriptors available.	Not available when Descriptor Ring processing is disabled.
13	CNTXT	Processing of a Descriptor written to the Command Queue has finished	Asserted directly after the result Descriptor is written. Asserted only when the Packet Descriptor Ring is disabled.
12	CMD	Processing of a Descriptor written to the Command Queue has finished	Asserted directly after the result Descriptor is written. Asserted only when the Packet Descriptor Ring is disabled.
11	DATAO	Packet Engine is ready to *return* data fragment of size	
10	DATAI	Packet Engine is ready to *receive* data fragment of size	
9	PEERR	Packet Engine processing error	
8	UDMS	User-initiated Master DMA transaction complete	
7	DMAAV	User DMA command registers are available	DMA transaction has been loaded and the holding registers are available.
6	DMAERR	Error occurred during a user-initiated DMA transfer	

Figure 17-67. Interrupt Unmasked and Masked Status Registers (CRYP0_INT_UNMSK/MSK)			
Bit	**Mnemonic**	**Description**	**Comments**
5	PKT	Packet Engine finished processing of a complete packet	All remaining data is read from the output buffer.
4	SLVERR	Illegal transfer to the PLB Slave requested	
3	MSTERR	Timeout occurred on a PLB Master transfer request	
2	TRNG	TRNG finished generation of a True Random Number	
1	PRNG	PRNG finished generation of a Pseudo Random Number	
0	PKA	PKA finished an operation	

17.2.64 Interrupt Mask Register (CRYP0_INT_EN)

This register configures the interrupt mask for the UIC. Writing 1 will enable the interrupt source and 0 will disable it. When an interrupt source is disabled the matching interrupt is cleared in the CRYP0_INT_MSK register.

Address: 0x0 E015 00A8

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-68. Interrupt Mask Register (CRYP0_INT_EN)			
Bit	**Mnemonic**	**Description**	**Comments**
32:16		Reserved	
15	DESC	Command to the Packet Engine to fetch the next Packet Descriptor from the PDR.	Applicable only when the Packet Fetch Engine has been enabled with a non-0 CRYP0_PE_RING_S register setting.
14	DESCRD	Specified number of Descriptors (1 – 63) have completed processing. Also occurs if a Descriptor has just been processed and there are no more input Descriptors available.	Not available when Descriptor Ring processing is disabled.
13	CNTXT	Processing of a Descriptor written to the Command Queue has finished	Asserted directly after the result Descriptor is written. Asserted only when the Packet Descriptor Ring is disabled.
12	CMD	Processing of a Descriptor written to the Command Queue has finished	Asserted directly after the result Descriptor is written. Asserted only when the Packet Descriptor Ring is disabled.
11	DATAO	Packet Engine is ready to *return* data fragment of size	
10	DATAI	Packet Engine is ready to *receive* data fragment of size	
9	PEERR	Packet Engine processing error	

Figure 17-68. Interrupt Mask Register (CRYP0_INT_EN) (continued)

Bit	Mnemonic	Description	Comments
8	UDMS	User-initiated Master DMA transaction complete	
7	DMAAV	User DMA command registers are available	DMA transaction has been loaded and the holding registers are available.
6	DMAERR	Error occurred during a user-initiated DMA transfer	
5	PKT	Packet Engine finished processing of a complete packet	All remaining data is read from the output buffer.
4	SLVERR	Illegal transfer to the PLB Slave requested	
3	MSTERR	Timeout occurred on a PLB Master transfer request	
2	TRNG	TRNG finished generation of a True Random Number	
1	PRNG	PRNG finished generation of a Pseudo Random Number	
0	PKA	PKA finished an operation	

17.2.65 Interrupt Configuration Register (CRYP0_INT_CFG)

This register configures the interrupt type which will be provided to the UIC.

Configuring the Interrupt output type for *Pulse* causes the interrupt signal to pulse low for two clock cycles when activated. When set for *Level*, the interrupt signal is set low until cleared by the processor (that is, it will follow the bit in the Masked Status Register). For the PLB bus, this is typically set to Level.

Address: 0x0 E015 00AC

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-69. Interrupt Configuration Register (CRYP0_INT_CFG)

Bit	Mnemonic	Description	Comments
32:2		Reserved	
1	IPCLR	Pulse interrupt clear 0 Interrupt must be reset by processor 1 Reset pulse interrupt	Used only when bit 0 is 1.
0	IHOT	Interrupt output type 0 Level 1 Pulse	

17.2.66 Interrupt Force Descriptor Read Register (CRYP0_INT_DESRD)

This register allows the processor to force the packet engine to read the next descriptor or block of descriptors. The data contents of the write operation are ignored. Any write to this register will cause the descriptor fetch. This register is only applicable when the External Descriptor Ring Handler is enabled (PE ring size is non-0).

If the Host reads this register, it returns zeros.

Address: 0x0 E015 00B0

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Note:

Figure 17-70. Interrupt Force Descriptor Read Register (CRYP0_INT_DESRD)

Bit	Mnemonic	Description	Comments
32:0		Don't care	

17.2.67 Interrupt Descriptor Count Register (CRYP0_INT_DESCT)

This register configures the number of Descriptors that must be completed before issuing a PE Descriptor Done interrupt. Valid settings range from 1 – 63.

Note: If the Packet Engine has processed at least one Descriptor and then has no valid Descriptor in its internal Command Queue, it issues a P/E Descriptor Done interrupt immediately.

Address: 0x0 E015 00B4

Size (bits): 32

Access: R/W

Reset: 0x0000 0001

Figure 17-71. Interrupt Descriptor Count Register (CRYP0_INT_DESCT)

Bit	Mnemonic	Description	Comments
32:6		Reserved	
5:0	DCCNT	Descriptor Complete Count	A setting of 0 is invalid and will be interpreted as a 1.

Example:

Assume the CRYP0_INT_DESCT register is set to 0x0004. If three packets arrive and are queued on the PDR, the security processor processes them and populates three result descriptors on the RDR. However, an interrupt will *not* be generated, since the counter is waiting until a fourth packet is processed. If some time passes before a fourth packet arrives, the three completed packets stall in the result queue without the processor being aware of them. The processor can predict this scenario and poll for Result Descriptors without waiting for an interrupt.

17.2.68 Device Control Register (CRYP0_DC_CTRL)

This register contains the device control information.

Address: 0x0 E016 0080

Size (bits): 32

Access: R/W

Reset: 0x0001 0001

Figure 17-72. Device Control Register (CRYP0_DC_CTRL)

Bit	Mnemonic	Description	Comments
32:18		Reserved	
17	TRNGE	TRNG Enable 0 TRNG is held in reset 1 Enable TRNG	It is useful to hold the TRNG in reset to conserve power if this function is not being used.
16	PKAE	PKA Enable 0 PKA held in reset 1 Enable PKA	It is useful to hold the PKA in reset to conserve power if this function is not being used.
15:1		Reserved	
0	TDESE	Triple-DES Enable 0 Attempts to execute a Triple-DES operation in the PE result in a Prohibited Algorithm status error 1 Enable Triple-DES encryption algorithm	

17.2.69 Device ID Register (CRYP0_DC_DEVID)

This register contains the device and vendor ID information.

Address: 0x0 E016 0084

Size (bits): 32

Access: Read only

Reset: 0x16AE 0094

Figure 17-73. Device ID Register (CRYP0_DC_DEVID)

Bit	Mnemonic	Description	Comments
31:16	VID	Vendor ID 0x16AE	
15:0	DID	Device ID 0x0094.	

17.2.70 Device Information Register (CRYP0_DC_DEVINF)

This register provides an indication of the security function features available in this instatiation. If a feature is enabled in this security function version, the corresponding bit is set to 1. If the feature is disabled, the corresponding bit is 0. Security function features labeled (yes) are available in this revision of the core.

Address: 0x0 E016 0088

Size (bits): 32

Access: Read only

Reset: 0x00C8 3722

Figure 17-74. Device Information Register (CRYP0_DC_DEVINF)

Bit	Mnemonic	Description	Comments
31:24		Reserved	
23	TRNG	True Random Number Generator (yes)	
22	PKA	Public Key Accelerator (yes)	
21	EMB	External Memory DMA Interface (no)	
20	SARAM	On chip SA RAM (no)	
19	PRNG	Pseudo Random Number Generator (yes)	
18:17		Reserved	
16	DFLT	Deflate Compression (no)	
15:14		Reserved	
13	SHA1	SHA 1 Hash (yes)	
12	MD5	MD5 Hash (yes)	
11		Reserved	
10	AES	AES (Rijndael) Encryption (yes)	
9	ARC4	ARC4 Encryption (yes)	
8	DES	DES/Triple-DES encryption (yes)	
7:4	MAJ	Revision Upper and lower nibbles define the Major and Minor Revisions of the security function.	The default revision is 2.2.
3:0	MIN		

17.2.71 DMA Source Address Register (CRYP0_DMA_USRC)

This register contains the DMA source address.

Address: 0x0 E016 0094

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-75. DMA Source Address Register (CRYP0_DMA_USRC)

Bit	Mnemonic	Description	Comments
31:0		DMA Source Address	

17.2.72 DMA Destination Address Register (CRYP0_DMA_UDST)

This register contains the DMA destination address.

Address: 0x0 E016 0098

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-76. DMA Destintion Address Register (CRYP0_DMA_UDST)

Bit	Mnemonic	Description	Comments
31:0		DMA Destination Address	

17.2.73 DMA Command Register (CRYP0_DMA_UCMD)

This register specifies the transfer length and the Bus IDs for the data source and destination. After writing this register, the transaction will be queued into the 4-channel DMA controller and is executed when the DMA scheduler passes control to the DMA channel.

Address: 0x0 E016 009C

Size (bits): 32

Access: R/W

Reset: 0x8000 0000

Figure 17-77. DMA Command Register (CRYP0_DMA_UCMD)

Bit	Mnemonic	Description	Comments
31:16		Reserved	
15:14	SRC	Transfer direction 0110 From external memory to the security function 1001 From the security function to external memory All other bit combinations are reserved.	
13:12	DST		
11:0	TXL	Transfer Length Data length in bytes for the DMA transaction	Valid values range from 1 – 4095.

17.2.74 DMA Configuration/Status Register (CRYP0_DMA_CFG)

The register is used to specify the maximum burst transfer size settings for master and slave and enabling incremental and locked transfers on the PLB.

Address: 0x0 E016 00D4

Size (bits): 32

Access: R/W

Reset: 0x0000 0003

Figure 17-78. DMA Configuration/Status Register (CRYP0_DMA_CFG)

Bit	Mnemonic	Description	Comments
31:10		Reserved	
9:8	PLBP	PLB priority 00 Very low 01 Low 10 High 11 Very high	Priority of PLB requests by the PLB master interface to the PLB arbiter
7:5		Reserved	
4	SREQ	Secondary request 0 Master does no secondary requests 1 NA This instantiation is fixed at SREQ = 0.	Read only
3:2		Reserved	
1:0	MBS	Maximum burst size in bytes at the PLB 00 32 (2 beats on the PLB) 01 64 (4 beats on the PLB) 10 128 (8 beats on the PLB) 11 256 (16 beats on the PLB) This instantiation is fixed at MBS = 11.	Read only When there is less data available then the maximum burst size, the length of the burst is less than the maximum burst size. Any requested transfers larger than this size will be broken up into multiple PLB Master burst transfers of this size or less.

17.2.75 PRNG Status Register (CRYP0_PRNG_STAT)

The register contains the status of the PRNG. Result Ready is mode dependent and is set only in Manual mode (CRYP0_PRNG_CTRL[AUTO] = 0). Busy is mode independent.

Address: 0x0 E017 0000

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-79. PRNG Status Register (CRYP0_PRNG_STAT)

Bit	Mnemonic	Description	Comments
31:2		Reserved	
1	RRDY	Result Ready 0 PRNG enabled for the next operation 1 Valid PRN available in the result register.	Set only when the PRNG is controlled by software, and CRYP0_PRNG_CTRL[AUTO] = 0.
0	BSY	Busy PRNG is busy 0 Valid PRN available in the output register. 1 CRYP0_PRNG_CTRL[E] = 1	This bit is also set when in Auto mode (CRYP0_PRNG_CTRL[AUTO] = 1) and the PRNG is busy In Auto mode this bit is cleared after a number has been generated, and the PRNG is waiting for the PE to use the PRN.

17.2.76 PRNG Control Register (CRYP0_PRNG_CTRL)

The register contains control settings for generation of Preens.

Address: 0x0 E017 0004

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-80. PRNG Control Register (CRYP0_PRNG_CTRL)

Bit	Mnemonic	Description	Comments
31:3		Reserved	
2	R128	Result 128 Enables the generation of 128-bit Pseudo-Random Numbers. 0 PRNG fills only the lower 64 bits of the result register (CRYP0_PRNG_RES0). 1 PRNG fills both 64-bit result registers, starting with the lower 64 bits (CRYP0_PRNG_RES0) and ending with the upper 64 bits (CRYP0_PRNG_RES1). This option reduces the performance by half.	A read always returns the last written value.
1	AUTO	Auto Enables the automatic generation of PRNs. 0 PRNG stops the automatic generation of a PRNs at completion of the PRNG algorithm. Can be cleared only when CRYP0_PRNG_STAT[BSY] = 0. 1 PRNG starts the automatic generation of 128-bit PRNs when CRYP0_PRNG_CTRL[E] = 1. This is a continuous operation until CRYP0_PRNG_CTRL[AUTO] = 0.	This mode can be used for generating IV's for the DES and AES algorithms. The length of the IV is automatically adjusted for the algorithm. The PRNG is under control of the PE and generates a new IV when required. A read always returns the last written value.
0	E	Enable Enables the generation of a new PRN 0 PRNG has started an operation and CRYP0_PRNG_CTRL[AUTO] = 0. 1 PRNG starts the generation of a PRN.	A write with zero has no effect. Always read as zero.

17.2.77 PRNG Seed Value L/H Registers (CRYP0_PRNG_SDL/H)

These registers contains the 64-bit secret seed value that is seeded once after reset and then automatically updated with the output of the third Triple-DES operation.

Address: 0x0 E017 0008 – CRYP0_PRNG_SDL
0x0 E017 000C – CRYP0_PRNG_SDH

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-81. PRNG Seed Value L/H Registers (CRYP0_PRNG_SDL/H)

Bit	Mnemonic	Description	Comments
31:0		Seed Value 0:31 – CRYP0_PRNG_SDL 63:32 – CRYP0_PRNG_SDH	

17.2.78 PRNG Key x Low/High Registers (CRYP0_PRNG_KxL/H)

There are two key register pairs numbered 0 and 1 (x). These four registers contain the 64-bit cryptograhic keys used for Triple-DES operations. Bit 31:0 are in the low (L) register while bits 63:32 are in the high (H) register.

The DES keys are 64-bits long, and include eight parity-check bits (bits on positions 0, 8, 16, 24, 32, 40, 48 and 56). The KEY registers are implemented using a 56-bit maximum-length LFSR implementation, ignoring the parity bits. The KEY registers are updated (LFSR changes to next value) after every PRNG operation. These registers should *not* contain all zeros.

Address: 0x0 E017 0010 – CRYP0_PRNG_K0L
0x0 E017 0014 – CRYP0_PRNG_K0H
0x0 E017 0018 – CRYP0_PRNG_K1L
0x0 E017 001C – CRYP0_PRNG_K1H

Size (bits): 32

Access: Write only

Reset: 0x0000 0000

Figure 17-82. PRNG Key x Low/High Registers (CRYP0_PRNG_KEYx_L/H)

Bit	Mnemonic	Description	Comments
31:0		PRNG Key Key bits 63:32 are in CRYP0_PRNG_KxH. Key bits 31:0 are in CRYP0_PRNG_KxL.	If the processor reads these registers, they return an undefined value.

17.2.79 PRNG Result x Registers (CRYP0_PRNG_RSx)

These registers contain two generated PRNs. A pair of result registers are concatenated to form one unique 128-bit result. The contents of the result registers are valid when CRYP0_PRNG_STAT[BSY] = 0 or the PRGN interrupt signal is high.

Address: 0x0 E017 0020 – CRYP0_PRNG_RS0
0x0 E017 0024 – CRYP0_PRNG_RS1
0x0 E017 0028 – CRYP0_PRNG_RS2
0x0 E017 002C – CRYP0_PRNG_RS3

Size (bits): 32

Access: Read only

Reset: 0x0000 0000

Figure 17-83. PRNG Result x Registers (CRYP0_PRNG_RSx)

Bit	Mnemonic	Description	Comments
31:0		PRNG Result Bits 31:0 are in CRYP0_PRGN_RS0 Bits 63:32 are in CRYP0_PRGN_RS1 Bits 95:64 are in CRYP0_PRGN_RS2 Bits 127:96 are in CRYP0_PRGN_RS3	

17.2.80 PRNG LFSR Low/High Registers (CRYP0_PRNG_LFL/H)

These registers contain a unique input DT as plain text input for the first Triple-DES operation. The register is implemented using a 64-bit maximum length LFSR. The register is updated (LFSR changes to next value) after every PRNG operation. These registers should *not* contain all zeros.

Address: 0x0 E017 0030 – CRYP0_PRNG_LFL
0x0 E017 0034 – CRYP0_PRNG_LFH

Size (bits): 32

Access: R/W

Reset: 0x0000 0000

Figure 17-84. PRNG LFSR Low/High Registers (CRYP0_PRNG_LFL/H)

Bit	Mnemonic	Description	Comments
31:0		CRYP0_PRNG_LFH – 32 most significant state bits [63:32] of the LFSR CRYP0_PRNG_LFL – 32 least significant state bits [31:0] of the LFSR	Reading these registers return the current value of the LFSR.

17.3 Buffer Interface

The security function has three buffers for use in processing data streams. Each of the buffers is described in detail in the following sections.

17.3.1 ARC4 Buffer

The ARC4 RAM buffer is used to store the pre-processed key that initializes the ARC4 function. Consists of a 256 Bytes write-only buffer. The packet input data is written here to be transferred to the Packet Engine.

The buffer is located at 0x0 E011 0700 – 0x0 E011 07FF. In Direct Host DMA mode, the processor must start writing from the base address (0x0 E011 0700) and increment the address pointer for each write.

A read from any address in the address range of the ARC4 Buffer returns an undefined value.

17.3.2 Input buffer

This is the location where input data is written into the security function in the Host DMA mode. The processor monitors the available space in the input buffer by means of the CRYP0_PE_DMA_ST register (see page 324).

This is a 2KB write-only buffer. The packet input data is written here to be transferred to the Packet Engine. The input buffer is located at 0x0 E0118000 – 0x0 E011 87FF.

In Direct Host DMA mode, the processor must start writing from the base address and increment the address pointer for each write. If a packet exceeds 2KB, the buffer will not be large enough and part of the buffer must be reused (wrapped back to the base address). The processor must monitor the CryptCore Input Size field (CRYP0_PE_DMA_ST[SEIS]) to keep track of the empty bytes available. To make data transfer from the processor memory to the Packet Engine more efficient, the processor can make use of Input Request Active, bit 10, in the CRYP0_PE_DMA_ST register. This bit indicates that the input buffer has N words free for write, where N is specified by the Packet Engine in the Input Threshold field (bits 9:0) of the CRYP0_PE_IO_THR register.

Example:

Assume a packet of 2400 bytes must be written to the input buffer, and the Input Threshold is set to 128 bytes. The processor starts writing to the input buffer at the base address 0x0 E0118000 and keeps incrementing the address by 4 for every word write until 2048 bytes are written. Then the processor reads the CryptCore Input Size field in the CRYP0_PE_DMA_ST register to find out how many bytes are available in the input buffer. When space is available, the address is wrapped and the next 400 bytes are written from address 0x0 E0118000 and onwards.

A read from any address in the address range of the input buffer returns an undefined value.

17.3.3 Output Buffer

This is the location (0x0 E011C000 – 0x0 E011C7FF) where output data is read from the security function in the Host DMA mode. The processor monitors the available bytes in the output buffer by means of the CRYP0_PE_DMA_ST register (see page 324).

It consists of a 2KB read-only buffer. The packet output data is read from here to be transferred to the processor memory. The processor must start reading from the base address (0x0 E011 C000) and increment the address pointer for each read. If a packet exceeds 2KB, the buffer will not be large enough and part of the buffer must be reused (wrapped back to the base address). The processor must monitor the CryptCore Output Size field in the CRYP0_PE_DMA_ST register to keep track of the filled bytes available in the Output Buffer. To make data transfer from the Packet Engine to the Host Memory more efficient, the processor can make use of the Output Request

Active, bit 11 in the CRYP0_PE_DMA_ST register. This bit indicates that the Output Buffer contains N words available for read, where N is specified by the Output Threshold field (bits 25:16) in the CRYP0_PE_IO_THR register.

A write to any address in the address range of the Output Buffer has no effect.

17.3.4 Public Key Memory (PKM)

RAM for Public Key input and output vectors. This memory is located at 0x0 E014 1000 – 0x0 E0141 FFF

18. KASUMI Algorithm

The built-in security feature (PPC440EPx-S and PPC440GRx-S) utilizes the KASUMI algorithm. The KASUMI algorithm is provided by chip circuitry attached to and accessed through the 128-bit PLB bus. The following sections provided detailed information on the functions and programming required to utilize the KASUMI function.

18.1 Features

The KASUMI function includes the following features:

- Key scheduling hardware
- *f8* confidentiality and *f9* integrity algorithm support
- Automatic data padding mechanism for f9 algorithm
- KASUMI encryption and decryption modes
- 32-bit slave interface
- Fully synchronous to the PLB clock

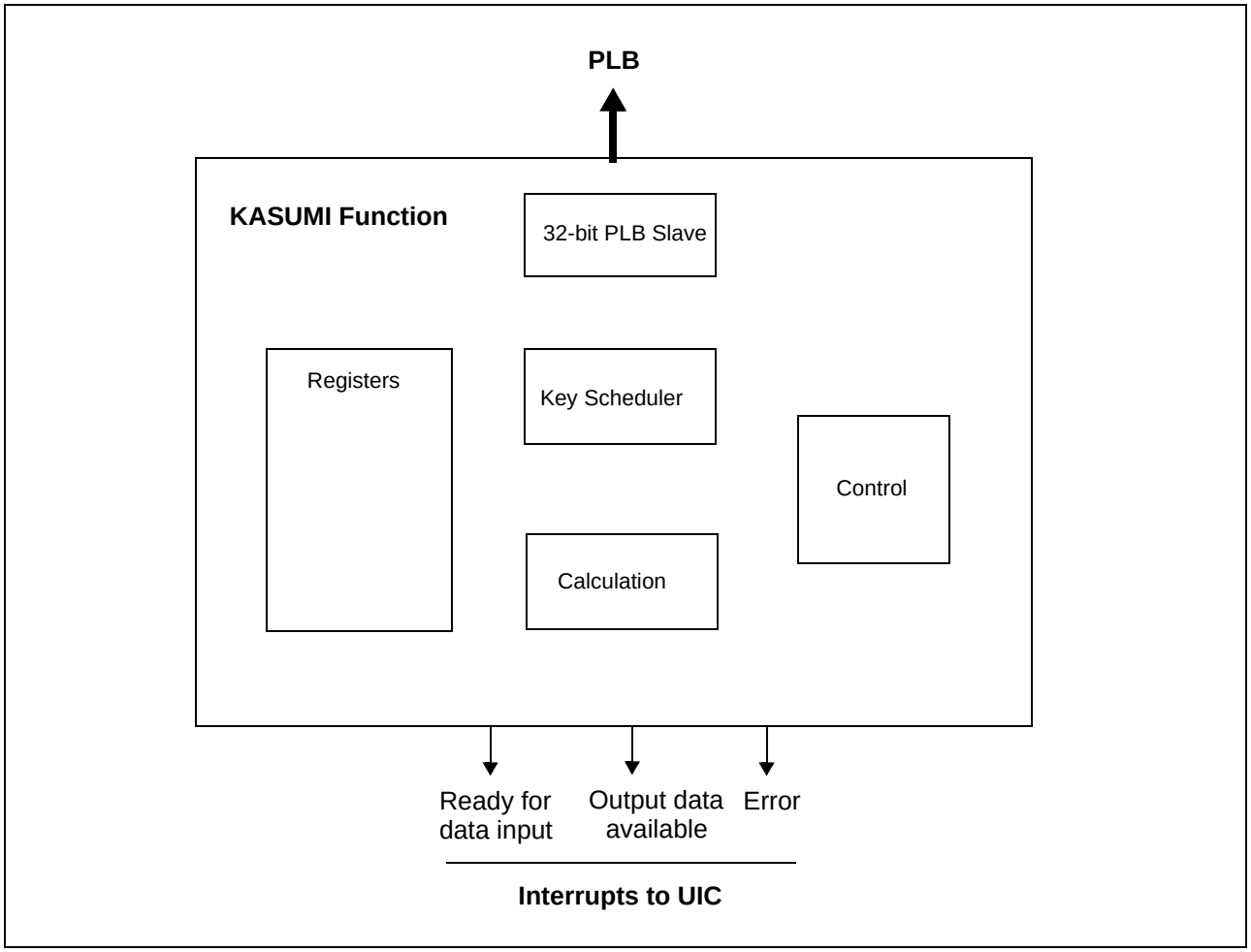
18.2 References

1. *Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 1: f8 and f9 Specification, V5.0.0*, June 2002
2. *Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 2: KASUMI Specification*, V5.0.0, June 2002

18.3 Block Diagram

The KASUMI function hardware is organized as shown in *Figure 18-1*.

Figure 18-1. KASUMI Block Diagram



18.4 Functional Description

The following sections provide a description of the functional characteristics of the KASUMI function.

18.4.1 General Processing

The KASUMI function can operate in three modes:

- KASUMI mode (encryption and decryption);
- *f8* mode;
- *f9* mode.

In general, a plain message data is parsed and sequentially fed into KASUMI in 64-bit blocks. The processing of one 64-bit block takes eight rounds. A round takes one clock cycle.

18.4.1.1 KASUMI Mode

In KASUMI mode, the 64-bit plain text data blocks are encrypted or decrypted under a 128-bit key, which is programmed into the KASUMI function. During each round, the Round Keys are derived from this initial key. After eight processing rounds, the KASUMI encrypted data block is stored in the output data register.

When performing encryption, plain text data is transformed into KASUMI cipher text data. When performing decryption, the KASUMI cipher text data is transformed back into the original plain text data.

In order to process data in KASUMI mode, the following KASUMI function registers must be programmed:
- KASU0_KEY0:3
- KASU0_CTRL[KEYAV] = 1
- MODE register by:
  - Selecting encrypt or decrypt mode (KASU0_MODE[CRYPT])
  - Selecting KASUMI mode (KASU0_MODE[MDKAS] = 1
- KASU0_CTRL[MODAV] = 1
- KASU0_DATAIN0:1
- KASU0_CTRL[DATAAV] = 1

KASUMI will start processing right after KASU0_CTRL[DATAAV] is programmed.

To process any additional message data in this session, write the message data in blocks of 64 bits into the data registers followed by setting KASU0_CTRL[DATAAV] =1. The KASUMI function remains in KASUMI mode until the mode register is programmed again.

Note: In this sequence, KASU0_CTRL can be programmed only once at the end on the sequence by simultaneously setting the KEYAV, MODAV and DATAAV fields.

More information about KASUMI mode can be found in Reference 2.

18.4.1.2 f8 Mode

In *f8* mode, the 64-bit message data blocks are transformed into 64-bit output data blocks under the control of a 128-bit Confidentiality Key. The total message length can be up to 2^{16} = 65536 bits.

In order to process data in *f8* mode, the following KASUMI registers must be programmed:
- KASU0_KEY0:3
- KASU0_CTRL[KEYAV] = 1
- KASU0_COUNT
- KASU0_CTRL[CNTAV] = 1
- KASU0_CONFIG
- KASU0_CTRL[CFGAV] = 1
- MODE register by:
  - Selecting encrypt mode (KASU0_MODE[CRYPT] = 1)
  - Selecting *f8* mode (KASU0_MODE[MDF8] = 1)
- KASU0_CTRL[MODAV] = 1
- KASU0_DATAIN0:1
- KASU0_CTRL[DATAAV] = 1

KASUMI starts processing immediately after KASU0_DATAIN and KASU0_CTRL[DATAAV] are programmed.

To process additional message data in this session, write the message data in blocks of 64 bits into the data registers followed by setting KASU0_CTRL[DATAAV] =1. KASUMI remains in *f8* mode until the mode register is programmed again.

Note: In this sequence, the KASU0_CTRL register can be programmed only once at the end on the sequence by simultaneously setting the KEYAV, CNTAV, CFGAV, MODAV and DATAAV fields.

More information about the *f8* algorithm can be found in Reference 1.

18.4.1.3 f9 Mode

In *f9* mode, the message data is processed in 64-bit blocks under control of a 128-bit Integrity Key. The result, after all message data has been processed, is a 32-bit Message Authentication Code (MAC). The total length of the message data can be up to 2^{16} = 65536 bits.

In order to process data in *f9* mode, the following KASUMI function registers must be programmed:

- KASU0_KEY0:3
- KASU0_CTRL[KEYAV] = 1
- KASU0_COUNT
- KASU0_CTRL[CNTAV] = 1
- KASU0_FRESH
- KASU0_CTRL[FSHAV] = 1
- KASU0_CONFIG
- KASU0_CTRL[CFGAV] = 1
- MODE register by:
  - Selecting encrypt mode (KASU0_MODE[CRYPT] = 1)
  - Selecting *f9* mode (KASU0_MODE[MDF9] = 1)
- KASU0_CTRL[MODAV] = 1
- KASU0_DATAIN0:1
- KASU0_CTRL[DATAAV] = 1

KASUMI starts processing immediately after KASU0_CTRL[DATAAV] is programmed.

For processing of any additional message data for this session it is sufficient to write the message data in blocks of 64 bits into the data registers, followed by setting KASU0_CTRL[DATAAV] =1. KASUMI remains in *f9* mode until the mode register is programmed again.

Note: In this sequence, the KASU0_CTRL register can be programmed only once at the end on the sequence by simultaneously setting the KEYAV, CNTAV, FSHAV, CFGAV, MODAV and DATAAV fields.

The KASU0_CONFIG register does not have to be programmed at the start of the session, but can be programmed later during the *f9* session. The latest chance to program the KASU0_CONFIG register is at the time the last data block is written to the KASUMI function. KASUMI needs the information in KASU0_CONFIG to perform message padding. When the last message data has been received, the KASUMI function automatically adds padding bits after the last bit of the message, as required by the *f9* algorithm.

When all data has been processed, an *f9* termination cycle will be performed. The termination cycle takes eight clock cycles. When the termination cycle is finished, the 32-bit MAC is available in KASU0_DATAOUT.

More information about the *f9* algorithm can be found in Reference 1.

18.4.2 Flow Control

Two signals generated by the KASUMI function provide flow control information to the CPU. They are connected to UIC0. Flow control information is also available in the KASU0_STAT register.

An output error signal is connected to UIC1.

Table 18-1. KASUMI Flow Control Signals

Output Signal	Connected To	Description	KASU0_STAT
inbuf_rfd	UIC0	Indicates that the function is ready to receive new input information. It is a continuously active level.	KASU0_STAT[6:1] = 0
output_data_av	UIC0	Indicates that the function has finished processing a data block and that the output data is available in the KASU0_DATAOUTx registers.	KASU0_STAT[OBUFF] = 1
sl_err	UIC1	Indicates a PLB read or write transfer error (for example, byte or half-word transfers).	na

18.5 Register Interface

The following sections describe the KASUMI register interface in detail. All of the KASUMI registers are 32-bit MMIO registers.

.

Table 18-2. KASUMI Registers

Register Name	Description	Address	Access	Page
KASU0_DATAIN0, KASU0_DATAOUT0	Data Input 0, Data Output 0	0x0 E018 0000	Write Read	394
KASU0_DATAIN1, KASU0_DATAOUT1	Data Input 1, Data Output 1	0x0 E018 0004	Write Read	394
KASU0_CTRL, KASU0_STAT	Control, Status	0x0 E018 0008	Write Read	395
KASU0_MODE	Mode input	0x0 E018 000C	Write only	396
KASU0_KEY0	Key input 31:0	0x0 E018 0010	Write only	397
KASU0_KEY1	Key input 63:32	0x0 E018 0014	Write only	397
KASU0_KEY2	Key input 95:64	0x0 E018 0018	Write only	397
KASU0_KEY3	Key input 127:96	0x0 E018 001C	Write only	397
KASU0_COUNT	Count input	0x0 E018 0020	Write only	398
KASU0_CONFIG	Configuration input	0x0 E018 0024	Write only	398
KASU0_FRESH	Fresh input	0x0 E018 0028	Write only	399

18.5.1 Byte Ordering

KASUMI registers are implemented and described in little endian format. Therefore, bit 31 is the most significant bit (msb) and bit 0 is the least significant bit (lsb). Big endian software must consider this when accessing the registers. For example, to put the value 44332211 in a KASUMI 32-bit register, big endian software must either do a store (**stw**) of the value 11223344 or do a byte-reversed store (**stwbr**) of the value 44332211. In either case, the value 44332211 appears on the appropriate word of the PLB data bus. Refer to the *Storage Addressing* section of the *PPC440 Processor User's Manual* for a more detailed description of this process.

18.5.2 Data Input/Output Register (KASU0_DATAIN0, KASU0_DATAOUT0)

Address: 0x0 E018 0000

Size (bits): 32

Access: Read – KASU0_DATAOUT0
Write – KASU0_DATAIN0

Reset: 0x0000_0000

Figure 18-2. Data Input/Output0 Register (KASU0_DATAIN0, KASU0_DATAOUT0)

Bit	Mnemonic	Description	Comments
31:0		Data 31:0	See KASU0_DATAIN1, KASU0_DATAOUT1 registers for Data 63:32.

Note: For a write operation, the complete 64-bit input block must be written to KASU0_DATAINx for the next KASUMI operation. The write to the input data buffer can occur any time *inbuf_rfd* output is HIGH. For a read operation, the KASU0_DATAOUTx registers contain the 64-bit output block from the latest KASUMI operation.

18.5.3 Data Input/Output Register (KASU0_DATAIN1, KASU0_DATAOUT1)

Address: 0x0 E018 0004

Size (bits): 32

Access: Read – KASU0_DATAOUT1
Write – KASU0_DATAIN1

Reset: 0x0000_0000

Figure 18-3. Data Input/Output0 Register (KASU0_DATAIN1, KASU0_DATAOUT1)

Bit	Mnemonic	Description	Comments
31:0		Data 63:32	See KASU0_DATAIN0, KASU0_DATAOUT0 registers for Data 31:0.

Note: For a write operation, the complete 64-bit input block must be written to KASU0_DATAINx for the next KASUMI operation. The write to the input data buffer can occur any time *inbuf_rfd* output is asserted. For a read operation, the KASU0_DATAOUTx registers contain the 64-bit output block from the latest KASUMI operation.

18.5.4 Control/Status Register (KASU0_CTRL, KASU0_STAT)

Address: 0x0 E018 0008

Size (bits): 32

Access: Read – KASU0_STAT
Write – KASU0_CTRL

Reset: 0x0000_0000

Figure 18-4. Control/Status Register (KASU0_CTRL, KASU0_STAT)

Bit	Mnemonic	Description	Comments
31:7		Reserved	
6	FSHAV	KASU0_FRESH register data 0 Validation complete 1 Request validation	This bit is automatically cleared (set to 0) when the KASUMI function completes validation of the KASU0_FRESH register. This bit is 1 between the time it is set by the request and the time the KASUMI function validates the KASU0_FRESH register. Writing 0 to this bit has no effect.
5	CNTAV	KASU0_COUNT register data 0 Validation complete 1 Request validation	This bit is automatically cleared (set to 0) when the KASUMI function completes validation of the KASU0_COUNT register. This bit is 1 between the time it is set by the request and the time the KASUMI function validates the KASU0_COUNT register. Writing 0 to this bit has no effect.
4	CFGAV	KASU0_CONFIG register data 0 Validation complete 1 Request validation	This bit is automatically cleared (set to 0) when the KASUMI function completes validation of the KASU0_CONFIG register. This bit is 1 between the time it is set by the request and the time the KASUMI function validates the KASU0_CONFIG register. Writing 0 to this bit has no effect.
3	DATAAV	KASU0_DATAINx register data 0 Cipher operation started 1 Request to start cipher operation	Can be set as soon as KASUMI is ready for new input (i.e., when input_rfd is 1). This bit is automatically cleared (set to 0) when the KASUMI function has started processing the KASU0_DATAINx contents. This bit is 1 between the time it is set by the request and the time the KASUMI function starts processing the KASU0_DATAINx registers. Writing 0 to this bit has no effect.
2	KEYAV	KASU0_KEYx register data 0 Validation complete 1 Request validation	This bit is automatically cleared (set to 0) when the KASUMI function completes validation of the KASU0_KEYx registers. This bit is 1 between the time it is set by the request and the time the KASUMI function validates the KASU0_KEYx registers. Writing 0 to this bit has no effect.

Figure 18-4. Control/Status Register (KASU0_CTRL, KASU0_STAT) (continued)

Bit	Mnemonic	Description	Comments
1	MODAV	KASU0_MODE register data 0 Validation complete 1 Request validation	This bit is automatically cleared (set to 0) when the KASUMI function completes validation of the KASU0_MODE register. This bit is 1 between the time it is set by the request and the time the KASUMI function validates the KASU0_MODE register. Writing 0 to this bit has no effect.
0	OBUFF	Output buffers contain processed data 0 No data available 1 Results of the latest encryption/decryption operation is available	After retrieving data from the output registers, write 1 to this bit to clear it. This makes the output registers available again for writing by the KASUMI function. Writing 0 to this bit has no effect.

18.5.5 Mode Register (KASU0_MODE)

Address: 0x0 E018 000C

Size (bits): 32

Access: Write only

Reset: 0x0000_0000

Figure 18-5. Mode Register (KASU0_MODE)

Bit	Mnemonic	Description	Comments
31:4		Reserved	
3	MDF9	Select *f9* mode 0 Not *f9* mode 1 *f9* mode	Setting this bit selects the *f9* mode.
2	MDF8	Select *f8* mode 0 Not *f8* mode 1 *f8* mode	Setting this bit selects the *f8* mode.
1	MDKAS	Select KASUMI mode 0 Not KASUMI mode 1 KASUMI mode	Setting this bit selects the KASUMI mode.
0	CRYPT	Select type of cipher operation 0 Decryption 1 Encryption	This bit must be set to 1 (encryption) during *f8* and *f9* mode.

18.5.6 Key Registers (KASU0_KEYx)

The following four registers are loaded with the 128-bit cipher key.

Address: 0x0 E018 0010 (KASU0_KEY0)
0x0 E018 0014 (KASU0_KEY1)
0x0 E018 0018 (KASU0_KEY2)
0x0 E018 001C (KASU0_KEY3)

Size (bits): 32

Access: Write only

Reset: 0x0000_0000

Figure 18-6. Key Register (KASU0_KEY0)

Bit	Mnemonic	Description	Comments
31:0		KEY data 31:0	

Figure 18-7. Key Register (KASU0_KEY1)

Bit	Mnemonic	Description	Comments
31:0		KEY data 63:32	

Figure 18-8. Key Register (KASU0_KEY2)

Bit	Mnemonic	Description	Comments
31:0		KEY data 95:64	

Figure 18-9. Key Register (KASU0_KEY3)

Bit	Mnemonic	Description	Comments
31:0		KEY data 127:96	

18.5.7 Count Register (KASU0_COUNT)

This register is used only during *f8* and *f9* modes.

Address: 0x0 E018 0020

Size (bits): 32

Access: Write only

Reset: 0x0000_0000

Figure 18-10. Count Register (KASU0_COUNT)

Bit	Mnemonic	Description	Comments
31:0		Count 31:0	This register must be written with the 32-bit count value. The write can occur any time the *inbuf_rfd* signal is asserted.

18.5.8 Configuration Register (KASU0_CONFIG)

This register is used only during *f8* and *f9* modes.

Address: 0x0 E018 0024

Size (bits): 32

Access: Write only

Reset: 0x0000_0000

Figure 18-11. Configuration Register (KASU0_CONFIG)

Bit	Mnemonic	Description	Comments
31:16	Length	*f9* Message data length	Must be programmed to the correct value prior to writing the last data block during an *f9* session. The total message length can be up to 2^{16} = 65536 bits.
15:6		Reserved	
5:1	BEAR	*f8* bearer value	This field is used only during *f8* mode. Must be programmed to the desired value prior to starting an *f8* session.
0	DIR	Direction of *f8* or *f9* session	Must be set to the correct value prior to starting an *f8* session, or prior to writing the last data block during an *f9* session.

18.5.9 Fresh Register (KASU0_FRESH)

This register is used only during *f9* mode.

Address: 0x0 E018 0028

Size (bits): 32

Access: Write only

Reset: 0x0000_0000

Figure 18-12. Fresh Register (KASU0_FRESH)

Bit	Mnemonic	Description	Comments
31:0		Fresh 31:0	This register must be written with the 32-bit Fresh value. The write can occur any time the *inbuf_rfd* signal is asserted.

18.6 Programing Examples

The following pseudo code examples show the actions that are typically executed by the software to start a cipher session in KASUMI.

18.6.1 KASUMI Mode Example

```
// wait until input buffer available for writing by host
wait KASU0_STAT[6:1] == '000000' (or inbuf_rfd == '1')

write KASU0_KEY0
write KASU0_KEY1
write KASU0_KEY2
write KASU0_KEY3

write KASU0_DATAIN0
write KASU0_DATAIN1

write KASU0_MODE[MDKAS] == '1' // select KASUMI mode
write KASU0_MODE[CRYPT]// select encryption or decryption

write KASU0_CTRL[6:0]        // indicate available data, e.g. 0x0E if Mode,
                             // Key and Data were written

// wait until output data ready
wait KASU0_STAT[OBUFF] == '1' (or outbuf_data_av == '1')
```

```
read KASU0_DATAOUT0
read KASU0_DATAOUT1

write KASU0_CTRL[OBUFF] = '1' // indicate finished reading output data
```

18.6.2 *f8* Mode Example

```
// wait until input buffer available for writing by host
wait KASU0_STAT[6:1] == '000000' (or inbuf_rfd == '1')

write KASU0_KEY0
write KASU0_KEY1
write KASU0_KEY2
write KASU0_KEY3

write KASU0_COUNT
write KASU0_CONFIG

write KASU0_DATAIN0
write KASU0_DATAIN1

write KASU0_MODE[MDF8] == '1' // select f8 mode
write KASU0_MODE[CRYPT] == '1' // select encrypt mode

write KASU0_CTRL[6:0]          // indicate available data, e.g. 0x3E if Mode,
                               // Key, Data, Config and Count were written

// wait until input buffer available for writing by host
wait KASU0_STAT[6:1] == '000000' (or inbuf_rfd == '1')

// Place next data block in pipeline
write KASU0_DATAIN0
write KASU0_DATAIN1

write KASU0_CTRL[6:0]        // indicate available data, e.g. 0x08 if Data
                             // was written

// wait until output data ready
wait KASU0_STAT[OBUFF] == '1' (or outbuf_data_av == '1')

// Read 1st cipher result
read KASU0_DATAOUT0
read KASU0_DATAOUT1
```

```
write KASU0_CTRL[OBUFF] = '1'// indicate finished reading output data

// wait until output data ready
wait KASU0_STAT[OBUFF] == '1' (or outbuf_data_av == '1')

// Read 2nd cipher result
read KASU0_DATAOUT0
read KASU0_DATAOUT1

write KASU0_CTRL[OBUFF] = '1'  // indicate finished reading output data
```

18.6.3 f9 Mode Example

```
// wait until input buffer available for writing by host
wait KASU0_STAT[6:1] == '000000' (or inbuf_rfd == '1')

write KASU0_KEY0
write KASU0_KEY1
write KASU0_KEY2
write KASU0_KEY3

write KASU0_COUNT
write KASU0_FRESH
write KASU0_CONFIG

write KASU0_DATAIN0
write KASU0_DATAIN1

write KASU0_MODE[MDF9] == '1'        // select f9 mode
write KASU0_MODE[CRYPT] == '1'     // select encrypt mode

write KASU0_CTRL[6:0]        // indicate available data, e.g. 0x7E if Mode,
                             // Key, Data, Config, Count and Fresh were written

// wait until input buffer available for writing by host
wait KASU0_STAT[6:1] == '000000' (or inbuf_rfd == '1')

// Place next data block in pipeline
write KASU0_DATAIN0
write KASU0_DATAIN1
```

```
write KASU0_CTRL[6:0]    // indicate available data, e.g. 0x08 if Data
                         // was written

// wait until output data ready
wait KASU0_STAT[OBUFF] == '1' (or outbuf_data_av == '1')

// Read 32-bit MAC result
read KASU0_DATAOUT0

write KASU0_CTRL[OBUFF] = '1'  // indicate finished reading output data
```

Part IV. PPC440EPx/GRx Peripheral Functions and Interfaces

19. DDR SDRAM Controller

The Double Data Rate 2/1 (DDR2/1) SDRAM memory controller supports industry standard discrete devices that are compatible with both the DDR1 or DDR2 specifications. The correct I/O supply voltage must be provided for the two types of DDR devices: DDR1 devices require +2.5V and DDR2 devices require +1.8V.

Global memory timings, address and bank sizes, and memory addressing modes are programmable.

Features include:

- 32-bit memory interface for DDR1
- 32- or 64-bit memory interface for DDR2
- Programmable memory data path size of 64/70 bit memory data width or 32/40 memory data width.
- ECC functionality for single-bit error correction and double-bit error detection circuity. Programmable detection and correction of ECC events. Programmable removal of ECC storage.
- 2.6-GB/s peak data rate
- 4/8 bank support.
- Maximum capacity of 2GB
- Support for 128-Mb, 256-Mb, 512-Mb, and 1-Gb DDR1 or DDR2 devices, with CAS latencies of 2, 2.5, or 3
- Clock frequencies from 133MHz (266Mbps) to 166MHz (333Mbps) supported.
- Support for DDR266/333 and DDR2-266/333.
  - Faster parts may be used, but must be clocked no faster than 166MHz
  - Page mode accesses (up to 16 open pages) with configurable paging policy
- Programmable address mapping and timing
- Software initiated self-refresh
- Power management (self-refresh, suspend, sleep)
- One or two chip selects
- x8 and x16 architecture support.
- Fully pipe lined command, read and write data interfaces to the controller.
- Advanced bank look-ahead features for high memory throughput.
- Programmable register interface to control memory device parameters and protocols.
- Full initialization of memory on controller reset.
- Built-in adjustable Delay Compensation Circuitry (DCC) for reliable data send and capture timing.
- No support for additive latency.

19.1 Initialization Protocol

The controller is designed such that it requires a sequence for correct operation after all power to the chip and to the memory devices is stable. After reset, the controller must be initialized and it will then automatically initialize the memory devices. The procedure to initialize the controller after the chip reset is complete is as follows:

1. Issue write register commands to configure the SDRAM and system protocols as indicated by the SYS and MEM types. No particular order is required. Keep the START bit deasserted during this initialization step.
2. Issue write register commands to configure and calibrate the DCC delay lines, as indicated by the CAL type. Keep the START bit de-asserted during this initialization step.
3. Assert the START bit. This triggers the controller to execute the initialization sequence using the parameters written into the registers. The controller will automatically initialize the memory devices and lock the internal DCC. The Delay Locked Loop (DLL) will process and send a signal to the initialization block when it has locked. Until all of these steps are completed, subsequent commands are blocked.
4. Wait for the DCC master delay line to finish calibration. This will be indicated by DLLLOCKREG = 1 (see DDR0_17 register). Under normal circumstances, this should happen so quickly that the user will immediately see 1 after setting the START bit.

All registers must be properly set before setting the start bit.

When doing a warm reset after the core is already running, clear DDR0_02 and DDR0_registers before setting the SDR0_RST0[DMC] bit. Failure to do this results in improper operation. These registers are cleared automatically at chip power-up, so this applies *only* to warm reset.

19.2 Device Address Mapping

The memory controller automatically maps user addresses to the DRAM memory in a contiguous block. Addressing starts at user address 0 and ends at the highest available address according to the size and number of DRAM devices present. This mapping is accomplished by setting certain parameters in the configuration registers.

The mapping of the address space to the internal data storage structure of the DRAM devices is ultimately determined by the actual size of the DRAM devices. The size is stored in user-programmable registers that must be initialized at power up.

Note: The word *device* refers to a single DDR1 or DDR2 chip. The word *organization* refers to the way in which the device is addressed (# rows, # columns, width of data path - x8 or x16 bits). *Modules* are generally synonymous with DIMMs, and are made up of multiple devices, which are usually all of the same organization (except in some cases where the data uses x16 architecture and the ECC uses x8). *Memory* refers to the entire array of devices and modules.

In the case of 1-rank configurations, Chip Select 0 must be populated. In the case of 2-rank configurations, both ranks must have the same organization and timing parameters.

19.2.1 DDR SDRAM Address Fields

The system address map, which is linear, is mapped into the address fields of the DDR SDRAM devices in the following order (msb to lsb):

1. Chip Select
2. Row
3. Bank
4. Column
5. Data path.

Note: Only x8, x16 and x32 devices are supported. Device width x4 is *not* supported.

19.2.1.1 Default Address Structure

Since the memory controller is designed to support different chip and system architectures, the maximum address range (8GB) that the controller can support natively is actually greater than the PPC440EPx/GRx allows (2GB). This maximum space is determined by the formula:

$$\text{MaxMemBytes} = \text{Chip selects} \times 2^{\text{address}} \times \text{Number of banks} \times \text{Data path width in bytes}$$

where:

- Chip selects = 2
- Number of device address bits = 14 + 12 (Row + Column)
- Number of banks = 8
- Data path width in bytes = 8 bytes

The default mapping of the system address space into the DRAM device address fields for the controller is shown in *Table 19-1*. This map corresponds to the largest memory device the controller core will *theoretically* support, with 14 row address bits, 12 column bits, three bank bits and an 8-byte (64-bit) bus for three data path bits. It shows the relative position and maximum possible width for each field.

Table 19-1. Default Controller Memory Address Mapping

Bits	Address Component
32	Chip Select
31:18	Row
17:15	Bank
14:3	Column
2:0	Data Path

19.2.1.2 Memory Mapping to Address Space

Since the system limits the maximum installed memory to 2GB, at most 30 bits are used to address the memory, as opposed to the default of 32 bits shown above. Fortunately, the address mapping algorithm is very flexible, which allows the memory controller to function with a wide variety of memory device sizes and organizations. While the Chip Select field width is fixed at one bit (CS0/CS1), the widths of the other fields of the memory map must be programmed using the device address width parameters ADDR_PINS (see DDR0_42), COLUMN_SIZE (see DDR0_43), REDUC (see DDR0_43) and EIGHT_BANK_MODE (see DDR0_43). The parameters are determined by the size and organization of the installed memory devices. The relationships between the controller parameters and the device organization are summarized below in *Table 19-2*.

The maximum width for any field may not be exceeded, but can be reduced by selecting different values for the parameter settings. The hardware automatically assures that the address space is contiguous starting at 0 by shifting the fields appropriately to the right as a function of their widths.

The ADDR_PINS and COLUMN_SIZE parameters can each range from the maximum configured for the controller (shown above) to seven bits smaller than the maximum configured. It is assumed that the values in these parameters never exceed the maximum values configured. Memory composed of DDR2 devices of 1Gbit or greater, have eight banks; all others have four banks. The width of the data bus, 32 or 64 bits (not including ECC), is dependent on the system design.

Table 19-2. DDR address fields and controlling parameters

	Chip Select	Row	Bank	Column	Data Path
Maximum value or possible configurations	2	2^{14}	4/8	2^{12}	32/64 bits
Maximum address field width	1	14	3	12	2/3
Maximum field width parameter (read only) (see DDR0_02	MAX_CS_REG	MAX_ROW_REG	n/a	MAX_COL_REG	n/a
Control parameter and formula	CS_MAP[1:0] Each bit is set (independently) if respective chip select (1:0) is populated.	ADDR_PINS = MAX_ROW_REG - # row bits in DRAM device	EIGHT_BANK_MODE 0 = 4 banks 1 = 8 banks[a]	COLUMN_SIZE = MAX_COL_REG - # col bits in DRAM device	REDUC 0 = 64 bits 1 = 32 bits

a.DDR2 devices of 1Gbit or greater use 8 banks. All other DDR2, and all DDR1 devices use 4 banks.

The formulas for calculating ADDR_PINS and COLUMN_SIZE are as follows:

ADDR_PINS = MAX_ROW_REG - number of row bits in memory device

COLUMN_SIZE = MAX_COL_REG - number of column bits in memory device

where MAX_ROW_REG = 14 and MAX_COL_REG = 12

To illustrate a specific example, we start with the default shown in *Table 19-1*. If the memory controller is wired for two ranks (2 CS) on a 64-bit bus composed of devices with four banks,13 address pins, 10 column bits, the maximum accessible memory space would be 512 MB and the address map for this configuration would be as shown in *Table 19-3*. Note that address bits 29 through 32 are not used. These bits are ignored when generating the address to the DRAM devices.

Table 19-3. Alternate Controller Memory Address Mapping

Bits	Address Component
32:29	Not used
28	Chip Select
25:15	Row
14:13	Bank
12:3	Column
2:0	Data Path

The parameters would be:

ADDR_PINS = 14 – 13 = 1
COLUMN_SIZE = 12 – 10 = 2
CS_MAP = 3
EIGHT_BANK_MODE = 0
REDUC = 0

Note: The Chip Select, Row, Bank, and Column fields are used to address an entire memory word, and the Data Path bits are used to address individual bytes within that user word. For example, for a read starting at byte address 0x2, the Data Path bits must be defined as 0b010 in order to address this byte directly. Reads and writes are memory word-aligned if all the Data Path bits are 0.

19.2.1.3 DDR Address Parameters for Common Organizations

The following table gives the correct parameter values for ADDR_PINS and COLUMN_SIZE to program for the most commonly available organizations. The only difference between 32- and 64-bit widths is the total amount of memory; the parameters are the same for devices of identical organization.

Table 19-4. DDR Address Parameters for Common Organizations

Device width (bits)		x8	x8	x8	x16	x16	x16
Device size (Mbit)		256	512	1024	256	512	1024
32-bit memory bus	Total Memory 1CS (MB)	128	256	512	64	128	256
	Total Memory 2CS (MB)	256	512	1024	128	256	512
64-bit memory bus	Total Memory 1CS (MB)	256	512	1024	128	256	512
	Total Memory 2CS (MB)	512	1024	2048	256	512	1024
Control register parameters							
DDR2	ADDR_PINS	1	0	0	1	1	1
	COLUMN_SIZE	2	2	2	3	2	2
DDR1	ADDR_PINS	1	1	0	1	1	0
	COLUMN_SIZE	1	1	1	1	1	1

The table below is provided as reference for the organization of devices mentioned above:

Table 19-5. Row/Column Organization of Common DDR Devices

Size (Mbit)	Width (bits)	Parameter (# bits)	DDR1	DDR2
256	8	Row	13	13
		Column	10	10
	16	Row	13	13
		Column	9	9
512	8	Row	13	14
		Column	11	10
	16	Row	13	13
		Column	10	10
1024	8	Row	14	14
		Column	11	10
	16	Row	14	13[a]
		Column	10	10

a.Note that 1Gbit DDR2 parts have 8 banks.

19.3 Register Interface

The SDRAM controller registers are indirectly addressed DCRs. The indirect address in hexadecimal corresponds to the decimal register number as it appears in the register mnemonic (for example, the address of DCR_11 is 0x0B).

19.3.1 DCR Access

All DDR SDRAM configuration registers are accessed with the move-to-DCR (**mtdcr**) and move-from-DCR (**mfdcr**) instructions using indirect addressing. In indirect addressing, the **mtdcr** and **mfdcr** instructions access a given DDR SDRAM register by means of its address offset which is contained in the DDR0_CFGADDR register. The data for the specified register is moved to or moved from the DDR0_CFGDATA register as described below. The DDR0 configuration registers are listed in *Table 19-6*.

Table 19-6. DDR SDRAM Controller DCR Addresses

Mnemonic	Register	Address	Access
DDR0_CFGADDR	DDR SDRAM Controller Address Register	0x0010	Write only
DDR0_CFGDATA	DDR SDRAM Controller Data Register	0x0011	R/W

Figure 19-1. DDR SDRAM Configuration Address Register (DDR0_CFGADDR)

0:23			
24:31	DCRA	8-bit SDRAM DDR Register Offset Value	This value can range from 0x00 to 0x7F. Its contents are used as the indirect DCR address for accessing an DDR SDRAM register.

Figure 19-2. DDR SDRAM Configuration Data Register (DDR0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contains the value of the DDR SDRAM register as indicated by the DDR0_CFGADDR register.

19.3.2 SDRAM Device Control Registers

A DDR SDRAM DCR may contain multiple parameters, a single parameter, or partial data for a parameter. As a result, a read of or a write to a particular parameter may require multiple read or write commands to different register addresses.

While parameters can be of any size, each parameter is mapped to byte boundaries that will fit the entire parameter. Undefined bits are *Reserved*. Reserved fields will return 0 on all register reads.

Unless otherwise noted, the following applies to all register fields:
- Default setting is 0
- Range of allowed values is 0 to the maximum value that the number of bits in the field can define
- Access is Read/Write

Table 19-7. Parameter Size to Mapping Condition

Parameter Size (bits)	Mapping Size	Starting Address
1 to 8	1 byte	1 byte boundary
9 to 16	2 bytes	2 byte boundary
17 to 128	4 bytes	4 byte boundary

The following figures define all the of the DCRs provided by the DDR SDRAM function. Each of the fields defined in the registers represent an operational parameter. The parameters are one of five types as indicated:

Table 19-8. DDR SDRAM Parameter Catagories

Type	Description
MEM	These parameters are STATIC and are strictly a function of the type of memory connected, and include those given by the appropriate manufacturer's data sheet or by the number or width of memories installed.
SYS	These parameters are STATIC and are a function of the system design choice (memory map, termination values, etc.). In some cases, certain values are strongly recommended. In other cases, sometimes the type of memory installed affects the acceptable values; these are indicated by the notation SYS/MEM.
CAL	These parameters can be changed during the course of delay line training. In specific systems or configurations, it might be possible to determine static values based on deterministic timing.
CMD	These parameters are used to send specific commands to the DDR devices, or to start and stop the controller.
ERR	These parameters are related to error checking and reporting.

19.3.3 DCR Descriptions

The following figures provide a detailed description of the contents by field (parameter) of the DCRs. The parameter type as described in *Table 19-8* is provided. The type of access (Read only, Write only, or Read/Write) for each field is also indicated. Fields with no access indicator are Read/Write.

Most timing parameters can be taken directly from the DDR SDRAM manufacturer's datasheet. Where timing parameters have a corresponding JEDEC parameter of the same name, it is indicated by JEDEC; if the parameter name is different, it is indicated by JEDEC = name. Be careful to convert timing information from ns to cycles as appropriate. Some parameters are not usually indicated in timing tables. In this case, they are noted with SPEC = see DDR datasheet text.

Figure 19-3. DDR0_00 Register

0		Reserved	
1:7	INTACK	INT_ACK Clear mask of the INT_STATUS parameter	ERR Write only If set to 1, the corresponding bit in the INT_STATUS parameter is set to 0. Writing INT_ACK with 0 does not alter the corresponding bit in the INT_STATUS parameter. This parameter is always read back as 0.
8:15	INTSTA	INT_STATUS Status of the interrupts in the controller. When set to 1, bits 8:15 have the following meaning: 8 = Logical OR of bits 9:15. 9 = DRAM initialization complete. 10 = Multiple uncorrectable ECC events detected. 11 = Single uncorrectable ECC event detected. 12 = Multiple correctable ECC events detected. 13 = Single correctable ECC event detected. 14 = Multiple accesses outside the defined *physical* memory space detected. 15 = Single access outside the defined *physical* memory space detected.	ERR Read only The msb is bit 8 and it is set only the first time the initialization sequence is completed after core reset. Subsequent re-initialization does not set this bit again, unless there is an intervening core reset.
16		Reserved	
17:23	DLLINC	DLL_INCREMENT Number of elements to add to DLL_START_POINT when searching for lock	CAL The parameter must be non-zero. The suggested value is 0x19.
24		Reserved	
25:31	DLLSP	DLL_START_POINT Initial delay count when searching for lock in master DLL	CAL The parameter must be non-zero. Suggested value is 0x0A

Figure 19-4. DDR0_01 Register			
0:2		Reserved	
3:7	CSLO	PLB0_DB_CS_LOWER PLB DATABAHN CS lower address	SYS Must be set to 0x01.
8:10		Reserved	
11:15	CSUP	PLB0_DB_CS_UPPER PLB DATABAHN CS upper address	SYS Must be set to 0x00.
16:20		Reserved	
21:23	OORTYP	OUT_OF_RANGE_TYPE Type of command that caused an Out-of-Range interrupt. READ-ONLY 000 Write 001 Read 010 Masked write 011 Illegal 100 Wrap write 101 Wrap read 110 Illegal 111 Illegal	ERR Read only Logs the type of the first error encountered.
24:31	INTMSK	INT_MASK Mask for controller interrupt signals from the INT_STATUS parameter	ERR This mask is inverted and then logically ANDed with the outputs of the INT_STATUS parameter. This result is then ORed to produce the interrupt.

Figure 19-5. DDR0_02 Register			
0:5		Reserved	
6:7	MAXCS	MAX_CS_REG Maximum number of chip selects available	MEM Read only This value can be used to set the CS_MAP parameter. CS_MAP = MAX_CS_REG – number of ranks of memory connected to the controller The number of ranks of memory must be a power of 2. Default = 0x2 Range = 0x0–0x2
8:11		Reserved	
12:15	MAXCOL	MAX_COL_REG Maximum width of column address in DRAMs	MEM Read only This value can be used to set the COLUMN_SIZE field in DDR0_43. COLUMN_SIZE = MAX_COL_REG – number of column bits in memory device Default = 0xC
16:19		Reserved	
20:23	MAXROW	MAX_ROW_REG Maximum width of memory address bus (number of row bits)	MEM Read only This value can be used to set the ADDR_PINS parameter. ADDR_PINS = MAX_ROW_REG – number of row bits in memory device. Default = 0xE Range = 0x0–0x4
24:30		Reserved	
31	START	START Initiate command processing in the controller 0 = Controller is not in active mode 1 = Initiate active mode for the controller	CMD When set to 0, the controller does not issue any commands to the DRAM devices or respond to any signal activity except for reading and writing parameters. Set to 1, the controller responds to inputs. The user should not set the start bit until the controller is fully initialized. Once operational, the START bit should not be cleared during operation.

Sets latency from read command send to data receive from/to controller.

Figure 19-6. DDR0_03 Register

0:4		Reserved	
5:7	BSTLEN	BSTLEN Encoded burst length sent to DRAMs during initialization 001 = 2 words 010 = 4 words This is the only valid setting for DDR2 devices and is the recommended setting for both DDR1/DDR2 devices. 011 = 8 words All other settings are Reserved.	SYS/MEM
8:12		Reserved	
13:15	CAS	CASLAT Encoded CAS latency sent to DRAMs during initialization	MEM JEDEC = CL The binary value programmed into this parameter is dependent on the memory device, since the same CASLAT value may have different meanings to different memories. This will be programmed into the DRAM devices at initialization. The CAS encoding will be specified in the DRAM spec sheet, and should correspond to the CASLAT_LIN parameter in the DRAM mode register. For example, a CAS latency of 2.5 cycles may be encoded as 110 in CASLAT, and 101 in CASLAT_LIN.
16:19		Reserved	
20:23	CLATLN	CASLAT_LIN Sets latency from read command send to data receive from/to controller. 0000 - 0010 = Reserved 0011 = 1.5 cycles 0100 = 2 cycles 0101 = 2.5 cycles 0110 = 3 cycles 0111 = 3.5 cycles 1000 = 4 cycles 1001 = Reserved 1010 = 5 cycles 1011 - 1111 = Reserved	CAL JEDEC = CL Sets the CAS latency linear value in 1/2 cycle increments. This sets an internal adjustment for the delay from when the read command is sent from the controller to when data is received back. The window of time in which the data is captured is a fixed length. The CASLAT_LIN parameter adjusts the start of this data capture window. The CASLAT_LIN should correspond to the CASLAT. For example, a CAS latency of 2.5 cycles may be encoded as 110 in CASLAT, and 101 in CASLAT_LIN. **Note:** Not all linear values will be supported for the memory devices being used (refer to the specifications for the memory device).
24:27		Reserved	
28:31	INAREF	INITAREF Number of auto-refresh commands to execute during DRAM initialization.	MEM SPEC = see DDR data sheet text

Figure 19-7. DDR0_04 Register			
0:2		Reserved	
3:7	TRC	TRC DRAM TRC parameter in cycles	MEM JEDEC DRAM period between active commands for the same bank.
8:12		Reserved	
13:15	TRRD	TRRD DRAM TRRD parameter in cycles	MEM JEDEC DRAM activate-to-activate delay for different banks.
16:20		Reserved	
21:23	TRTP	TRTP DRAM TRTP parameter in cycles	MEM JEDEC DRAM read to precharge time.
24:31		Reserved	

Figure 19-8. DDR0_05 Register			
0:2		Reserved	
3:7	TMRD	TMRD DRAM TMRD parameter in cycles	MEM JEDEC DRAM mode register set command time.
8:12		Reserved	
13:15	TEMRS	TEMRS DRAM TEMRS parameter in cycles	MEM JEDEC = TMRD DRAM extended mode parameter set time.
16:19		Reserved	
20:23	TRP	TRP DRAM TRP parameter in cycles	MEM JEDEC DRAM pre-charge command time.
24:31	TRASMN	TRAS_MIN DRAM TRAS_MIN parameter in cycles	MEM JEDEC = TRAS DRAM minimum row activate time.

Figure 19-9. DDR0_06 Register			
0:6		Reserved	
7	WINTRP	WRITEINTERP Allow controller to interrupt write bursts to the DRAMs with a read command 0 = No support for read commands interrupting write commands. 1 = Support for read commands interrupting write commands.	MEM SPEC = see DDR data sheet text Some memory devices do not allow this functionality.
8:12		Reserved	
13:15	TWTR	TWTR DRAM TWTR parameter in cycles	MEM JEDEC Sets the number of cycles needed to switch from a write to a read operation, as dictated by the DDR SDRAM specification.
16:23	TDLL	TDLL DRAM TDLL parameter in cycles	MEM SPEC = see DDR data sheet text DRAM DLL lock time.
24		Reserved	
25:31	TRFC	TRFC DRAM TRFC parameter in cycles	MEM JEDEC DRAM refresh command time.

Figure 19-10. DDR0_07 Register

0:6		Reserved	
7	NCI	NO_CMD_INIT Disable DRAM commands until TDLL has expired during initialization 0 = Issue only REF and PRE commands during DLL initialization of the DRAM devices 1 = Do not issue any type of command during DLL initialization of the DRAM devices	SYS Should be set to 0.
8:10		Reserved	
11:15	TFAW	TFAW DRAM TFAW parameter in cycles	MEM JEDEC
16:22		Reserved	
23	ARM	AUTO_REFRESH_MODE Sets if automatic refresh is at next burst or next command boundary 0 = Issue refresh on the next DRAM burst boundary, even if the current command is not complete 1 = Issue refresh on the next command boundary	SYS/MEM If AUTO_REFRESH_MODE is set and a refresh is required to memory, the controller delays this refresh until the end of the current transaction (if the transaction is fully contained inside a single page), or until the current transaction hits the end of the current page. **Note:** For PLB devices using full data path (the REDUC parameter is set to 0), this parameter must be set to 1. Because of this, it is recommended that this bit always be set to 1 regardless of configuration.
24:30		Reserved	
31	AREFR	AREFRESH Initiate auto-refresh when specified by AUTO_REFRESH_MODE	CMD Write only If there are any open banks when this parameter is set, the controller automatically closes these banks before issuing the auto-refresh command. This parameter always reads back 0.

Figure 19-11. DDR0_08 Register			
0:4		Reserved	
5:7	WRLAT	WRLAT DRAM WRLAT parameter in cycles	MEM SPEC = see DDR data sheet text For DDR2, WRLAT = CL – 1 Defines the write latency from when the write command is issued to the time the write data is presented to the DRAM devices.
8:15	TCPD	TCPD DRAM TCPD parameter in cycles	MEM SPEC = see DDR datasheet text. If in doubt, set to 200. Clock enable to precharge delay time for the DRAM devices
16:22		Reserved	
23	DQSNEN	DQS_N_EN Single-ended or differential DQS pins 0 = Single-ended DQS signal from the DRAM 1 = Reserved	MEM Enables differential data strobe signals from the DRAM. Must be set to 0.
24:30		Reserved	
31	DDR2	DDRII_SDRAM_MODE Select DDR1 or DDR2 mode 0 = DDR1 mode 1 = DDR2I mode	MEM See also SDR0_DDRCFG.

Figure 19-12. DDR0_09 Register

0:7	OAPDC0	OCD_ADJUST_PDN_CS_0 OCD pull-down adjust setting for DRAMs for chip select 0 0:3 = OCD setting 0 = decrement 1 = increment 4:7 = Number of OCD adjust commands to issue	CAL The controller issues OCD adjust commands to the DRAM devices during power up.
8:13		Reserved	
14:15	RTT0	RTT_0 On-Die termination resistance setting for chip select 0 00 = Termination Disabled 01 = 75Ω 10 = 150Ω 11 = Reserved	SYS The memory controller can not be set for different termination values for each chip. The RTT_0 values are expected to be uniform for all chip selects. **Note:** 75Ω termination is the recommended setting.
16		Reserved	
17:23	WDSB	WR_DQS_SHIFT_BYPASS Fraction of a cycle of delay in the write data path in the controller when DLL is being bypassed	CAL Controls the amount of delay introduced in the write data path when the DLL is being bypassed. This delay is introduced in increments of an internal delay element, which can range between 40ps and 120ps depending on conditions, to ensure the correct capture of data internally in the I/O logic. This parameter must be non-zero.
24		Reserved	
25:31	WDS	WR_DQS_SHIFT Fraction of a cycle of delay in the write data path in the controller	CAL Controls the amount of delay introduced to the write data path in fractions of a cycle to ensure the correct capture of data internally in the I/O logic. The value is expressed in fractions of 1/128 of a cycle; e.g., 0100000 sets 32 * 1/128 or 1/4 of a cycle. This parameter must be greater than 1/2 cycle. It should nominally be 1/4 cycle earlier than DQS. For example: DQS_OUT_SHIFT – WR_DQS_SHIFT = 0x20 or WR_DQS_SHIFT = 0x5F

Figure 19-13. DDR0_10 Register			
0:14		Reserved	
15	WMDREG	WRITE_MODEREG Write EMRS data to the DRAMs.	CMD Write-only Supplies the EMRS data for each chip select to allow individual chips to set masked refreshing. When this parameter is set to 1, the mode parameter(s) within the DRAM devices are written. Each subsequent WRITE_MODEREG setting writes the EMRS register of the next chip select. This parameter will always read back as 0. The mode registers are automatically written at initialization of the controller. There is no need to initiate a mode register write after setting the START parameter unless some value in these registers needs to be changed after initialization.
16:21		Reserved	
22:23	CSMAP	CS_MAP Number of active chip selects used in address decoding 00 = No memory installed 01 = Rank 1 not present, rank 0 installed 10 = Rank 1 installed, rank 0 not present 11 = Rank 1 installed, rank 0 installed	MEM The user address chip select field will be mapped into the active chip selects indicated by this parameter in ascending order from lowest to highest. This allows the controller to map the entire contiguous user address into any group of chip selects. Bit 0 of this parameter corresponds to chip select 0.
24:26		Reserved	
27:31	OAPUC0	OCD_ADJUST_PUP_CS_0 OCD pull-up adjust setting for DRAMs for chip select 0 27 = OCD setting 0 = decrement 1 = increment 28:31 = Number of OCD adjust commands to issue	CAL The controller issues OCD adjust commands to the DRAM devices during power up.

Figure 19-14. DDR0_11 Register			
0:6		Reserved	
7	SREFR	SREFRESH Place DRAMs in self-refresh mode 0 = Disable self-refresh mode 1 = Initiate self-refresh of the DRAM devices	CMD The current burst for the current transaction (if any) completes, all banks are closed, the self-refresh command is issued to the DRAM, and the clock enable signal is deasserted. The system will remain in self-refresh mode until this parameter is set to 0. The DRAM devices return to normal operating mode after the self-refresh exit time (TXSR) of the device and any DLL initialization time for the DRAM is reached. The controller resumes processing of the commands from the interruption point. Before placing the DRAMs in self-refresh mode, the DDR0_01[CSLO] and DDR0_01[CSUP] fields must be cleared. Take care to restore these fields before resuming normal operation.
8:15	TXSNR	TXSNR DRAM TXSNR parameter in cycles	MEM JEDEC
16:23	TXSR	TXSR DRAM TXSR parameter in cycles	MEM JEDEC = TXSRD DRAM self-refresh exit time.
24:31		Reserved	

Figure 19-15. DDR0_12 Register			
0:28		Reserved	
29:31	TCKE	TCKE Minimum CKE pulse width in cycles	MEM JEDEC

Figure 19-16. DDR0_14 Register			
0:6		Reserved	
7	DLLBYP	DLL_BYPASS_MODE Enable the DLL bypass feature of the controller 0 = Normal operational mode. 1 = Bypass the DLL master delay line	CAL Defines the behavior of the DLL bypass logic. When set to 0, the parameters that control the various delay lines are DLL_DQS_DELAY_x, DQS_OUT_SHIFT_x and WR_DQS_SHIFT, which express the delay in terms of a fraction of a clock cycle as measured by the master delay line. When set to 1, the master DCC locking mechanism is bypassed, and the parameters that control the delay line are DLL_DQS_DELAY_BYPASS_x, DQS_OUT_SHIFT_BYPASS_x and WR_DQS_SHIFT_BYPASS, which express the delay in terms of absolute number of delay elements (which range between about 40ps and 120ps depending on the silicon manufacturing process, temperature and operating voltage. If the total delay time programmed into the delay parameters exceeds the number of delay elements in the delay chain, the delay is set to the maximum number of delay elements in the delay chain.
8:14		Reserved	
15	REDUC	REDUC Enable the half data path feature of the controller 0 = Standard operation using full 64/72-bit memory bus 1 = Memory data path width is 32/40 bits	MEM Choose between a 23/40 and 64/72 bit data bus. The entire PLB bus is used regardless of this setting.
16:22		Reserved	
23	RDIM	REG_DIMM_ENABLE Enable registered DIMM operation of the controller 0 = Normal operation 1 = Enable registered DIMM operation	MEM Controls the address and command pipeline of the controller.
24:31		Reserved	

Figure 19-17. DDR0_17 Register			
0		Reserved	
1:7	DDQSD0	DLL_DQS_DELAY_0 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 0 during reads	CAL Sets the delay for the DQS signal from the DDR SDRAM devices for slice_0. This delay is used center the edges of the DQS signal so that the read data will be captured in the middle of the valid window in the I/O logic. Delay is added in increments of 1/128 of the system clock (e.g., 0100000 sets 32 * 1/128 or 1/4 of a cycle). The parameter must be non-zero. The nominal value is 0x19 (1/4 cycle).
8:14		Reserved	
15	DLKREG	DLLLOCKREG DLL lock/unlock	CAL Read only
16		Reserved	
17:23	DLLLOK	DLL_LOCK Number of delay elements in master DLL lock	CAL Read only Automatically updated every time a refresh operation is performed.
24:31		Reserved	

Figure 19-18. DDR0_18 Register			
0		Reserved	
1:7	DDQSD4	DLL_DQS_DELAY_4 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 4 during reads	See DLL_DQS_DELAY_0.
8		Reserved	
9:15	DDQSD3	DLL_DQS_DELAY_3 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 3 during reads	See DLL_DQS_DELAY_0.
16		Reserved	
17:23	DDQSD2	DLL_DQS_DELAY_2 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 2 during reads	See DLL_DQS_DELAY_0.
24		Reserved	
25:31	DDQSD1	DLL_DQS_DELAY_1 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 1 during reads	See DLL_DQS_DELAY_0.

Figure 19-19. DDR0_19 Register			
0		Reserved	
1:7	DDQSD8	DLL_DQS_DELAY_8 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 8 during reads	See DLL_DQS_DELAY_0.
8		Reserved	
9:15	DDQSD7	DLL_DQS_DELAY_7 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 7 during reads	See DLL_DQS_DELAY_0.
16		Reserved	
17:23	DDQSD6	DLL_DQS_DELAY_6 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 6 during reads	See DLL_DQS_DELAY_0.
24		Reserved	
25:31	DDQSD5	DLL_DQS_DELAY_5 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 5 during reads	See DLL_DQS_DELAY_0.

Figure 19-20. DDR0_20 Register			
0		Reserved	
1:7	DDDB3	DLL_DQS_DELAY_BYPASS_3 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 3 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.
8		Reserved	
9:15	DDDB2	DLL_DQS_DELAY_BYPASS_2 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 2 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.
16		Reserved	
17:23	DDDB1	DLL_DQS_DELAY_BYPASS_1 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 1 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.
24		Reserved	
25:31	DDDB0	DLL_DQS_DELAY_BYPASS_0 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 0 during reads when DLL is being bypassed	CAL Sets the delay for DQS from the DDR SDRAM devices for slice 0 for reads when the DLL is being bypassed This delay is used center the edges of the DQS signal so that the read data will be captured in the middle of the valid window in the I/O logic. Delay is added in increments of an internal delay element, which can range between about 40ps and 120ps depending on conditions. The parameter must be non-zero.

Figure 19-21. DDR0_21 Register			
0		Reserved	
1:7	DDDB7	DLL_DQS_DELAY_BYPASS_7 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 7 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.
8		Reserved	
9:15	DDDB6	DLL_DQS_DELAY_BYPASS_6 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 6 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.
16		Reserved	
17:23	DDDB5	DLL_DQS_DELAY_BYPASS_5 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 5 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.
24		Reserved	
25:31	DDDB4	DLL_DQS_DELAY_BYPASS_4 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 4 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.

Figure 19-22. DDR0_22 Register

0:5		Reserved	
6:7	CTLRAW	CTRL_RAW ECC error checking and correcting control 00 = ECC not being used 01 = ECC checking is on, but no correction is attempted 10 = No ECC RAM storage available 11 = ECC checking and correcting on	SYS
8		Reserved	
9:15	DQSOSB	DQS_OUT_SHIFT_BYPASS Fraction of a cycle to delay the write DQS signal to the DRAMs during writes when DLL is being bypassed	CAL Controls the amount of delay in increments of an internal delay element, which can range between about 40ps and 120ps depending on conditions, when the DLL is being bypassed. This delay is introduced to ensure correct data capture in the I/O logic. The parameter must be non-zero.
16		Reserved	
17:23	DQSOSH	DQS_OUT_SHIFT Fraction of a cycle to delay the write DQS signal to the DRAMs during writes when DLL is being bypassed	CAL Controls the amount of delay in fractions of a cycle introduced into the DQS signal on writes to ensure correct data capture in the DRAMs (e.g., 0100000 sets 32 * 1/128 or 1/4 of a cycle). The parameter must be non-zero. Nominal value is 0x7F.
24		Reserved	
25:31	DDDB8	DLL_DQS_DELAY_BYPASS_8 Fraction of a cycle to delay the DQS signal from the DRAMs for slice 8 during reads when DLL is being bypassed	See DLL_DQS_DELAY_BYPASS_0.

Figure 19-23. DDR0_23 Register			
0:5		Reserved	
6:7	ORMC0	ODT_RD_MAP_CS0 ODT Chip Select 0 map for reads. Determines which chip(s) will have termination when a read occurs on chip 0 6 = CS0 has active ODT termination when chip select 0 is performing a read 7 = CS1 has active ODT termination when chip select 1 is performing a read	SYS Only one chip select (and therefore 1 bit) may be set at any time. It is strongly recommended that all bits be set to 0 for reads. **Example:** If the system consists of 2 chip selects and ODT_RD_MAP_CS0 is set to 0b10, then when CS0 is performing a read, CS1 will have active ODT termination. If ODT_RD_MAP_CS0 is set to 0b01, then CS0 would be active instead.
8:15	ECSYN	ECC_C_SYND Syndrome for correctable ECC event	ERR Read only
16:23	EUSYN	ECC_U_SYND Syndrome for uncorrectable ECC event	ERR Read only
24:30		Reserved	
31	FWC	FWC Force a write check. XOR XOR_CHECK_BITS with ECC code and write to memory 0 = No action 1 = Force a write check	CMD Write only This parameter always reads back as 0. **Note:** Once the next write operation is complete, the controller automatically clears this bit.

Figure 19-24. DDR0_24 Register			
0:5		Reserved	
6:7	RTTPT	RTT_PAD_TERMINATION Set termination resistance in controller pads 00 = Termination Disabled 01 = 75 Ω 10 = Reserved 11 = Reserved	SYS Recommended setting is 01 (75 Ω).
8:13		Reserved	
14:15	OWMC1	ODT_WR_MAP_CS1 ODT Chip Select 1 map for writes. Determines which chip(s) will have termination when a write occurs on chip 1	See ODT_WR_MAP_CS0.
16:21		Reserved	
22:23	ORMC1	ODT_RD_MAP_CS1 ODT Chip Select 1 map for writes. Determines which chip(s) will have termination when a read occurs on chip 1	See ODT_RD_MAP_CS0.
24:29		Reserved	
30:31	OWMC0	ODT_WR_MAP_CS0 ODT Chip Select 0 map for writes. Determines which chip(s) will have termination when a write occurs on chip 0 01 = CS0 will have active ODT termination when CSx is performing a write 10 = CS1 will have active ODT termination when CSx is performing a write	SYS Only one chip select (and therefore 1 bit) may be set at any time. **Example:** If the system consists of two chip selects and ODT_WR_MAP_CS0 is set to 0b10, then when CS0 is performing a write, CS1 has an active ODT termination. If ODT_WR_MAP_CS0 is set to 0b01, then CS0 has an active ODT termination. It is strongly recommended that for the case of 1 chip select, all bits be set to 1 for that chip select for writes, i.e.: If CS0 is the only slot populated, then ODT_WR_MAP_CS0 = 0b01 and ODT_WR_MAP_CS1 = don't care. If CS1 is the only slot populated, then ODT_WR_MAP_CS1 = 0b10 and ODT_WR_MAP_CS0 = don't care. In the case of 2 chip selects, the bits corresponding to the chip select *not* addressed should be set to 1: ODT_WR_MAP_CS0 = 0b10 ODT_WR_MAP_CS1 = 0b01

Figure 19-25. DDR0_25 Register			
0:16	VER	VERSION Controller version number	Read only Default = 0x2041 Range = 0x0000–0x2041
16:21		Reserved	
22:31	OORLEN	OUT_OF_RANGE_LENGTH Length of command that caused an Out-of-Range interrupt	Read only ERR

Figure 19-26. DDR0_26 Register			
0:15	TRASMAX	DRAM TRAS_MAX parameter in cycles	MEM JEDEC = TRAS DRAM maximum row active time.
16:17		Reserved	
18:31	TREF	DRAM TREF parameter in cycles	MEM JEDEC Average number of DRAM cycles between refresh commands.

Figure 19-27. DDR0_27 Register			
0:1		Reserved	
2:15	EMRSDT	EMRS_DATA Extended mode register data written during initialization or when WRITE_MODEREG is 1	CMD Consult the DRAM specification for the correct settings for this parameter. When in DDR2 mode, the controller uses the value in RTT_0 and multiplexes it into the EMRS data being written during the initialization routine. It is not necessary to program the EMRS_DATA parameter with the same information.
16:31	TINIT	TINIT DRAM TINIT parameter in cycles	MEM SPEC = see DDR data sheet text DRAM initialization time.

Figure 19-28. DDR0_28 Register			
0:1		Reserved	
2:15	EMRS3D	EMRS3_DATA EMRS3 Data written during DDRII initialization.	CMD
16:17		Reserved	
18:31	EMRS2D	EMRS2_DATA EMRS2 Data written during DDRII initialization.	CMD

Figure 19-29. DDR0_31 Register			
0:15		Reserved	
16:31	XORCB	XOR_CHECK_BITS Value to XOR with generated ECC codes for forced write check.	ERR Result is written into memory as the new check value.

Figure 19-30. DDR0_32 Register			
0:31	OORALO	OUT_OF_RANGE_ADDR31:0 Address of command that caused an Out-of-Range interrupt	ERR Read only This address is the controller address, not the PLB system address.

Figure 19-31. DDR0_33 Register			
0:30		Reserved	
31	OORAUP	OUT_OF_RANGE_ADDR32 Address of command that caused an Out-of-Range interrupt	ERR Read only See OUT_OF_RANGE_ADDR31:0.

Figure 19-32. DDR0_34 Register			
0:31	EUALO	ECC_U_ADDR31:0 Address of uncorrectable ECC event	ERR Read only The controller pads this parameter with zeros for any address bits not used by the controller. This address is the controller address, not the PLB system address.

Figure 19-33. DDR0_35 Register			
0:30		Reserved	
31	EUAUP	ECC_U_ADDR32 Address of uncorrectable ECC event	ERR Read only See ECC_U_ADDR31:0.

Figure 19-34. DDR0_36 Register			
0:31	EUDLO	ECC_U_DATA31:0 Data associated with uncorrectable ECC event	ERR Read only

Figure 19-35. DDR0_37 Register			
0:31	EUDUP	ECC_U_DATA63:32 Data associated with uncorrectable ECC event	ERR Read only

Figure 19-36. DDR0_38 Register			
0:31	ECALO	ECC_C_ADDR31:0 Address of correctable ECC event	ERR Read only The controller will pad this parameter with zeros for any address bits not used by the controller. This address is the controller address, not the PLB system address.

Figure 19-37. DDR0_39 Register			
0:30		Reserved	
31	ECAUP	ECC_C_ADDR32 Address of correctable ECC event	ERR Read only See ECC_C_ADDR31:0.

Figure 19-38. DDR0_40 Register			
0:31	ECDLO	ECC_C_DATA31:0 Data associated with correctable ECC event	ERR Read only

Figure 19-39. DDR0_41 Register			
0:31	ECDUP	ECC_C_DATA63:32 Data associated with correctable ECC event	ERR Read only

Figure 19-40. DDR0_42 Register			
0:4		Reserved	
5:7	APIN	ADDR_PINS Difference between the maximum number of address pins available (14) and the number being used	MEM. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter See *Table 19-2*.
8:27		Reserved	
28:31	CLGATE	CASLAT_LIN_GATE Adjusts data capture gate open by half-cycles 0000 - 0010 = Reserved 0011 = 1.5 cycles 0100 = 2 cycles 0101 = 2.5 cycles 0110 = 3 cycles 0111 = 3.5 cycles 1000 = 4 cycles 1001 = Reserved 1010 = 5 cycles 1011 - 1111 = Reserved	CAL JEDEC=CL Adjusts the data capture gate open time by 1/2 cycle increments. This parameter is programmed differently than CASLAT_LIN when there are fixed offsets in the flight path between the memory and the controller for clock gating. When CASLAT_LIN_GATE is a larger value than CASLAT_LIN, the data capture window will become shorter. A CASLAT_LIN_GATE value smaller than CASLAT_LIN might have no effect on the data capture window, depending on the fixed offsets in the chip and the board. Should be set to the same value as CASLAT_LIN (DDR0_03[CLATLN])

Figure 19-41. DDR0_43 Register			
0:4		Reserved	
5:7	TWR	TWR DRAM TWR parameter in cycles	MEM DRAM write recovery time
8:11		Reserved	
12:15	APREBT	AUTO_PRE_BIT Location of the automatic precharge bit in the DRAM address	MEM The PPC440EPx/GRx does not support precharge. The location of the bit in the address must be correctly identified for the controller to function properly. This bit is usually 9 or 10 on the address bus. See manufacturer's datasheet.
16:20		Reserved	
21:23	COLSIZ	COLUMN_SIZE Difference between the maximum number of 12 column pins available and the number being used	MEM The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter. See *Table 19-2*.
24:30		Reserved	
31	BNK8	EIGHT_BANK_MODE Number of banks on the DRAM(s) 0 = Memory devices have 4 banks 1 = Memory devices have 8 banks	MEM SPEC = see DDR data sheet text.

Figure 19-42. DDR0_44 Register			
0:23		Reserved	
24:31	TRCD	TRCD DRAM TRCD parameter in cycles	MEM JEDEC Delay time between active commands to read or write.

19.4 Clocking

The memory controller is configured with a clocking system that propagates a clock from the controller to the memory devices. The greatest challenge with this clock forwarding scheme is the management of the clock skew for the round-trip data transfer.

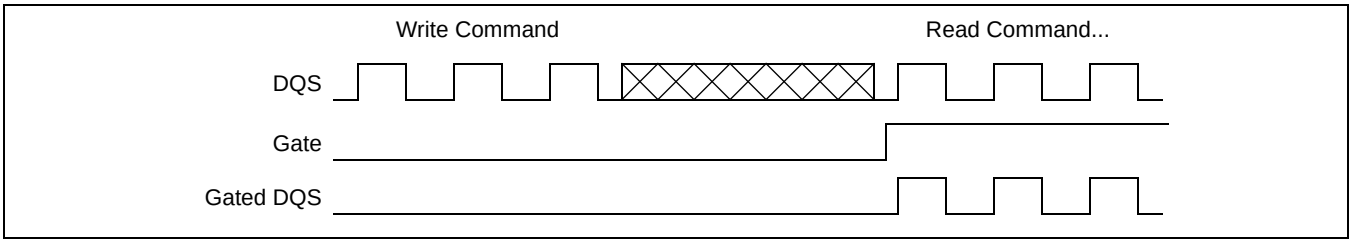
In such a system, the command and address for the transaction are sent from the controller coincident with the falling edge of the controller clock. In the case of registered DIMMs, where the clock, command, and address signals have roughly the same pad and flight delays to travel to the memory, the rising edge of the clock at the memory is centered with the command and address signals, allowing reliable capture. Otherwise, with unbuffered memory, care must be taken when designing the clocking system on a board. See the *PowerPC 440EPx and PowerPC 440GRx Embedded Processor Data Sheet* for more details.

To ensure proper behavior, two user-programmable parameters control the write path: DQS_OUT_SHIFT and WR_DQS_SHIFT. These two parameters set the delay for the external DQS0:7 outputs and for the MemData00:63, DM0:8, and ECC0:7 outputs, respectively. These parameters should be set such that the DQS0:7 outputs are in phase with MemClkOut and that the MemData00:63/DM0:8/ECC0:7 outputs are 1/4 cycle ahead of of the DQS0:7 outputs

.For the read path, the flight paths must be taken into consideration. There is a certain time lag from when the clock is sent from the controller to when the data and DQS signals are received at the controller from the memory. Since the DQS from the memory will be sent coincident with the data, and the data must be captured reliably, the DQS signal must be delayed so that it is centered in the data valid window (nominally 1/4 cycle).

The DQS bus is a bidirectional bus that is driven by the controller on writes and the memory on reads. When neither device is driving the bus, DQS will remain in a high-impedance state. However, DQS is only relevant to the controller during reads in order to capture valid data. For this reason, the DQS signal from memory is gated inside the controller so that it is ignored at all other times. Gating of the DQS signal is shown in *Figure 19-43*.

Figure 19-43. DQS Gating



When to start gating DQS depends on the design itself, the flight time of the clock to memory, and the flight time of the data and DQS to the controller, as follows:

If the round trip time is between 1/2 cycle and 1.5 cycles, set the CASLAT_LIN parameter equal to the CASLAT parameter. Based on estimated timing (to be confirmed in characterization), if the board routing guidelines mentioned in the PPC440EPx/GRx data sheet are observed, the round trip time for the PPC440EPx/GRx should always be in this range.

If the round trip time is less than 1/2 cycle, set the CASLAT_LIN parameter one value less (which translates to 1/2 cycle) than the CASLAT parameter to open the gate 1/2 cycle sooner.

If the round trip time is longer than 1.5 cycles, set the CASLAT_LIN parameter one value more (which translates to 1/2 cycle) than the CASLAT parameter to open the gate 1/2 cycle later.

In addition, the CASLAT_LIN_GATE parameter controls the opening of the gating signal. Nominally, CASLAT_LIN_GATE should have the same value as the CASLAT_LIN parameter. However, to accommodate the skew of the memory devices, it might be necessary to open the gate a 1/2 cycle sooner or later. Adjusting the value of CASLAT_LIN_GATE modifies the gate opening by this factor.

19.5 Error Checking and Correction (ECC)

The ECC circuitry incorporates these high level features:

- The ability to correct single-bit errors and detect double-bit errors for a 64-bit data field.
- The ability to detect and correct errors in both the check code and the memory.
- Storing the failing address, data and syndrome information on the first occurrence of a single-bit (correctable) and a double-bit (un-correctable) error. Subsequent ECC events are indicated by a user-accessible status parameter within the controller..
- The ability to force bad ECC to be written into the memory for diagnostic or flagging purposes.
- A programmable option in the controller to ignore or detect all ECC events, and to correct or not correct ECC errors.

The memory controller is configured to employ ECC coverage for each 64-bit word in the data path to memory. Eight bits of ECC storage are used for each 64-bit data word. The ECC circuitry is capable of single-bit error correction and double-bit error detection.

19.5.1 Controlling ECC

ECC functionality is controlled through the CTRL_RAW parameter. This parameter is subdivided into two functions: Bit 24 controls the detection and reporting of ECC events and bit 25 controls whether or not the data will be corrected, if possible, by the ECC circuitry. Table 49, "ECC Control Parameter Settings" shows the bit settings for CTRL_RAW.

Table 19-9. ECC Control Parameter Settings

CTRL_RAW	Description
00	ECC detection and correction is disabled. The INT_STATUS parameter and the ECC reporting parameters are never updated to reflect any ECC events in this mode.
01	Enable ECC detection only. ECC events are reported through processor interrupts, the INT_STATUS parameter, and the ECC reporting parameters. No correction is attempted for single-bit errors.

Table 19-9. ECC Control Parameter Settings

CTRL_RAW	Description
10	ECC DRAM devices are not wired to the PPC440EPx/GRx. Place all ECC data, data strobes and data mask drivers in output mode and drive a value of 0. This mode is used when the PPC440EPx/GRx is not connected to any ECC RAM storage.
11	ECC detection and correction is enabled. When errors are found, interrupts are asserted. The INT_STATUS parameter and the ECC reporting parameters are updated. Single-bit errors are corrected automatically by the controller.

When CTRL_RAW equals 0b01 or 0b11 and any ECC event is detected, the INT_SATUS parameter is updated to reflect the ECC event. The appropriate status bit in the INT_STATUS parameter will remains set until it is manually cleared by writing to the INT_ACK parameter. It is recommended that the user set the CTRL_RAW parameter to 0b00 before writing to the INT_ACK or INT_MASK parameters to ensure that no other ECC events are logged during the ECC interrupt routines that clear the status or mask parameters.

The controller supports 64-bit ECC check codes. These codes are applied to a contiguous 64-bit data field in the data path of the controller. ECC error checking is only performed on operations that involve a read of data. This includes all read operations, all masked write operations, and any write that does not start and end on an ECC word boundary. While the controller supports byte transactions of any length, the controller only calculates ECC on specific byte-aligned sections of the memory. For this reason, any write operation for a request of less than 64-bits (8 bytes) or any request that is not aligned to a 8-byte boundary is internally modified to a read/modify/write operation to allow for proper ECC check code calculation.

19.5.2 ECC Error Types

An ECC error is defined as correctable or uncorrectable. A correctable error is a single-bit error that will be fixed by the controller automatically if the CTRL_RAW parameter is set to 0b11. An un-correctable error involves two or more bits and will not be modified. When an ECC error is detected, either the correctable or uncorrectable ECC parameters will be set with the address (ECC_C_ADDR, ECC_U_ADDR), data (ECC_C_DATA, ECC_U_DATA) and syndrome (ECC_C_SYND, ECC_U_SYND) that relate to the error. Refer to *Syndrome Codes* on page 438 for more information on syndromes.

In addition, when an error occurs, if enabled, an interrupt will be asserted for the read data that caused the ECC event.

The INT_STATUS parameter holds the status of any ECC events that have occurred. The parameter is cleared on reset of the controller. The logical OR of the bits of the INT_STATUS parameter is reflected in the controller after being passed through the INT_MASK parameter. *Table 19-10* describes the information in this parameter.

Table 19-10. ECC Error Parameter Description

Bit	Description
INT_STATUS (2)	Set to 1 if a correctable ECC event has been detected.
INT_STATUS (4)	Set to 1 if an un-correctable ECC event has been detected.
INT_STATUS (3)	Set to 1 if more than one correctable ECC event has been detected.
INT_STATUS (5)	Set to 1 if more than one un-correctable ECC event has been detected.
INT_MASK	Mask field (bits 5:2) to inhibit assertion of ECC interrupt signals. Refer to the full description of the INT_MASK parameter in *Register Interface* on page 410.

If an ECC event is detected on a read triggered by an unaligned write to the controller, the INT_STATUS parameter is set to the appropriate value. Single-bit errors are corrected during the write phase of the read/modify/write operation. Double-bit errors cause the ECC bits written into memory to indicate a double-bit error condition if this word is ever read again by the controller.

19.5.3 ECC and Read Operations

A read operation from memory returns 128 bits. The action of the controller on this read data is dependent on the value of the CTRL_RAW parameter:

- ECC detection and correction is disabled (CTRL_RAW is set to 0b00):

  The controller sends the read data to the data interface.

- Only ECC detection is on (CTRL_RAW is set to 0b01):

  The controller calculates the check code(s) for each ECC word and XOR the calculated value(s) with the value(s) returned with the data. If the XOR result is all zeros, then the memory value is correct. If the XOR returns any 1 results, then either a correctable or an uncorrectable error has occurred. The ECC parameters are updated with the relevant data. The incorrect data is sent to the data interface.

- ECC detection and correction is on (CTRL_RAW is set to 0b11):

  The controller detects the error when ECC detection is on. However, if the error was a single-bit (correctable), then the controller uses the syndrome code to determine which bit to change, correct the data, and send the corrected data to the data interface. Note that the incorrect bit is still stored in memory.

19.5.4 ECC and Write Operations

The behavior of a write command related to ECC operations depends entirely on how the byte address and length of the write aligns with ECC words. For write commands where both the starting and ending address of the write fall on ECC word boundaries, the controller calculates the ECC and then writes both the data and the ECC into memory as a normal write operation.

Masked write commands and any write commands whose starting and ending address do not fall on ECC word boundaries are not completed in the same way. Since the value of the check code depends on the entire ECC word, and only some of the bytes of the ECC word are new data, the controller must know the value of the other bytes in this ECC word to calculate the ECC check code. As a result, these types of operations are modified internally to Read/Modify/Write operations. A read/modify/write operation consists of these steps:

- The user word containing the write data address is read out.
- If ECC error detection is enabled, then the controller calculates the expected check code(s) and XOR the calculated check code(s) with the value(s) read from memory.
  - If the two check codes are identical, then there are no ECC errors.
  - If a correctable ECC error is detected, and ECC error correction is enabled, then the controller determines which bit to fix using the syndrome code, and correct the erring bit. If ECC error correction is disabled, then the error is not corrected.
  - If an uncorrectable ECC error is detected, then the controller automatically corrupts the check code for the entire user word. This ensures that a future access to this location will reveal an ECC error.
- The corrected or uncorrected user word is combined with the new write data. The controller calculates a new check code(s).
- The new check code(s) (accurate or corrupted) and the ECC words that were modified are written to memory. Even though the entire user word of 128 bits is available in the controller, only the ECC words and check codes that were changed are written back to memory. The remaining bits are masked out.

19.5.5 Syndrome Codes

Check code generation is the major value of ECC. The circuitry is used to confirm the validity of memory contents by performing an XOR operation on the check code from the memory and the calculated check code from the actual memory contents. The operation produces a syndrome code which is stored in the ECC_C_SYND or ECC_U_SYND parameters on an ECC operation. The syndrome code identifies which bit (on a single-bit error) of the memory data or the check code is incorrect. If a pattern in the syndrome does not select a particular data bit which must be flipped, then the error is multibit and therefore uncorrectable.

Table 19-11 displays the syndrome codes that correspond to single-bit error, no error, and multibit errors.

Table 19-11. 64-Bit ECC Syndrome Codes

Bit in Error	Code	Bit in Error	Code	Bit in Error	Code	Bit in Error	Code
Data 0	0XF4	Data 16	0XB5	Data 32	0x75	Data 48	0x34
Data 1	0XF1	Data 17	0XB0	Data 33	0x70	Data 49	0x31
Data 2	0XEC	Data 18	0XAD	Data 34	0X6D	Data 50	0X2C
Data 3	0XEA	Data 19	0XAB	Data 35	0X6B	Data 51	0X2A
Data 4	0XE9	Data 20	0XA8	Data 36	0X68	Data 52	0X29
Data 5	0XE6	Data 21	0XA7	Data 37	0X67	Data 53	0X26
Data 6	0XE5	Data 22	0XA4	Data 38	0X64	Data 54	0X25
Data 7	0XE3	Data 23	0XA2	Data 39	0X62	Data 55	0X23
Data 8	0XDC	Data 24	0X9D	Data 40	0X5E	Data 56	0X1C
Data 9	0XDA	Data 25	0X9B	Data 41	0X5B	Data 57	0X1A
Data 10	0XD9	Data 26	0X98	Data 42	0X58	Data 58	0X19
Data 11	0XD6	Data 27	0X97	Data 43	0X57	Data 59	0X16
Data 12	0XD5	Data 28	0X94	Data 44	0X54	Data 60	0X15
Data 13	0XD3	Data 29	0X92	Data 45	0X52	Data 61	0X13
Data 14	0XCE	Data 30	0X8F	Data 46	0X4F	Data 62	0X0E
Data 15	0XCB	Data 31	0X8A	Data 47	0X4A	Data 63	0X0B
Check 0	0x80	Check 2	0x20	Check 4	0x08	Check 6	0x02
Check 1	0x40	Check 3	0x10	Check 5	0x04	Check 7	0x01
No error		0x00		Multi-bit error		Any other 8 bit code	

19.5.6 Forcing an ECC Error Event

There are situations where the user might wish to force an ECC event to be triggered. This could be used for testing purposes to force an ECC error condition when no error occurs.In other instances, the user may wish to tag a particular memory location as erroneous. This might be used when an un-correctable error occurs on a read/ modify/write operation. This method corrupts the ECC check code in memory so that ECC errors will be forced on subsequent reads of this location.

The procedure for forcing an ECC event is as follows:

1. Set the CTRL_RAW parameter to 0b01. This will detect, but not correct, ECC events on reads.
2. Ensure that no writes to the controller are pending.
3. Write a value to the XOR_CHECK_BITS parameter that will trigger an ECC event once that word is read. Use the syndrome codes listed in *Syndrome Codes* on page 438 to set the XOR_CHECK_BITS parameter. Each byte of the XOR_CHECK_BITS controls the ECC event forcing for a separate user-word space. *Table 19-12* shows how each byte of the XOR_CHECK_BITS parameter maps to user word bits.

4. For example, to force a single-bit correctable error on bit 0 of the user-word space shown, write 0xF4 into that byte of the XOR_CHECK_BITS parameter. To force a multi-bit un-correctable error for the user-word space, write 0x03 into that byte of the XOR_CHECK_BITS parameter.
5. Assert the FWC parameter using the register interface.
6. Execute a write command to the controller for an aligned user word.
7. If the user wishes for the ECC event to be triggered immediately, then the following steps should also be completed.
8. Execute a read command for that same word.
9. A correctable ECC event will be triggered. The INT_STATUS parameter will be set to indicate an ECC error. The ECC_C_ADDR, ECC_C_DATA and ECC_C_SYND parameters will be set with the relevant information.

Once the first word is written to memory with the FWC parameter enabled, the *fwc* bit will be cleared by the controller. The next read operation to this address will then trigger the appropriate ECC event. This event will store the data, address, and syndrome bits of the event. Refer to *Table 19-12* for details on how to force a bad ECC syndrome for a particular piece of write data.

Table 19-12. Parameter XOR_CHECK_BITS *Mapping*

Mapping	Description
XOR_CHECK_BITS 15:8 map to user word 127:64 XOR_CHECK_BITS 7:0 map to user word 63:0	This value will be XORed with the generated ECC bits and then written to memory when the FWC bit is also set. When this user word is read back, an ECC event will be detected. For codes to force a particular event, refer to the ECC syndrome codes table.

19.5.7 Clearing a Reported ECC Event

To clear a reported ECC event, the user should follow these steps:
- Set CTRL_RAW parameter to 0b00. This prohibits any possible ECC events from being recorded during the recovery of the stored ECC event.
- Read the ECC data, address, and syndrome parameters to determine where the event occurred.
- Set the ECC bit in the INT_ACK parameter. This clears the ECC event and allows future events to be captured.
- Set the CTRL_RAW parameter back to its original value.

19.6 Delay Compensation Circuit

Due to the asynchronous nature of the DRAM devices, the timing requirements for capturing and receiving data between the PPC440EPx/GRx and the DRAM devices must be addressed in any memory controller design. The PPC440EPx/GRx controller contains a delay compensation circuit that, in conjunction with I/O cell circuitry, can be used to meet the timing requirements for the memory subsystem. The delay compensation circuit was designed with the following features:

- Programmable read DQS delay specified as a percentage of a clock cycle.
- Programmable write data and write DQS delays specified as percentages of a clock cycle.
- Delay compensation circuit re-sync circuitry activated during refresh cycles to compensate for temperature and voltage drift.
- Separate delay chains for each DQS signal from the DRAM devices.

The delay compensation circuitry relies on a master/slave approach. There is a master delay line which is used to determine how many delay elements constitute a complete cycle. This count is used, along with the programmable fractional delay settings, to determine the actual number of delay elements to program into the slave delay lines. The master and slave delay lines are identical. This approach allows the controller to observe a clock and then delay other signals a fixed percentage of that clock. The DCC logic does not actively generate clock signals.

The delay parameters are listed in *Table 19-13*. The total delay can be determined based on the following equation, where *param* is one of the parameters in the table:

delay = #delays in one cycle x param/128

Table 19-13. Delay Parameters

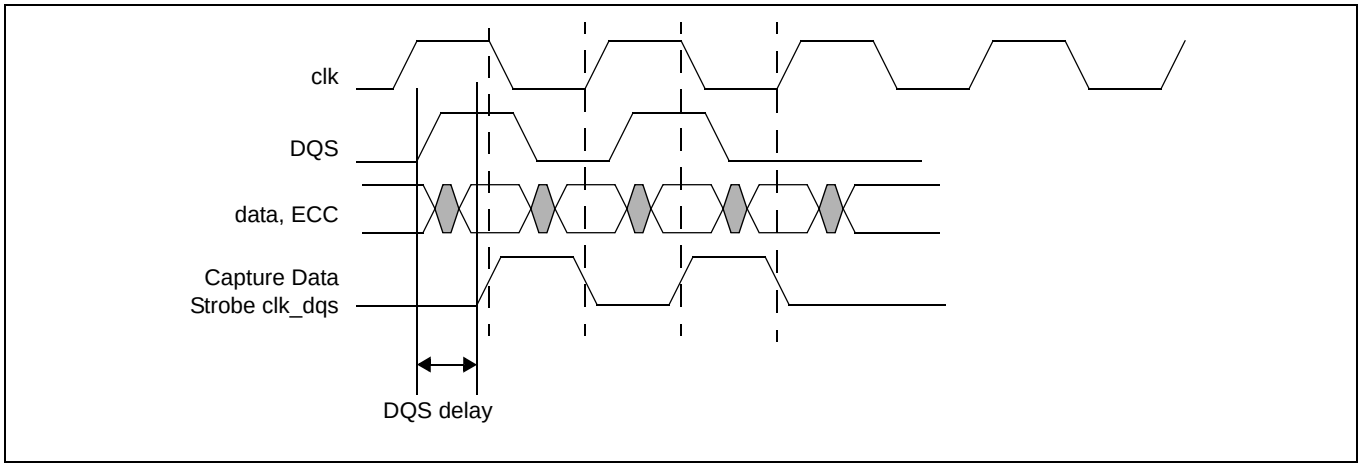
Operation	Parameter
Read	DLL_DQS_DELAY
Writes	DQS_OUT_SHIFT
Clock	WR_DQS_SHIFT

19.7 Read Data Capture

The read data capture logic is responsible for capturing the data and ECC outputs from the DRAM devices and passing the data back to the system clock domain.

Instead, a delayed version of the data strobe signal must be used to capture the data in the . The delay added to the data strobe signals should be such that the margin to capture the read data is maximized. Because the frequency of the data strobe signal is matched to the system clock, the delay is a relative number based on the period of the system clock. In the example shown in *Figure 19-44*, the delay is set to approximately 25% of the system clock. A delay compensation circuit in the will keep this relative delay constant so that the read data from the DRAM devices can be reliably captured.

Figure 19-44. DQS Read Timing



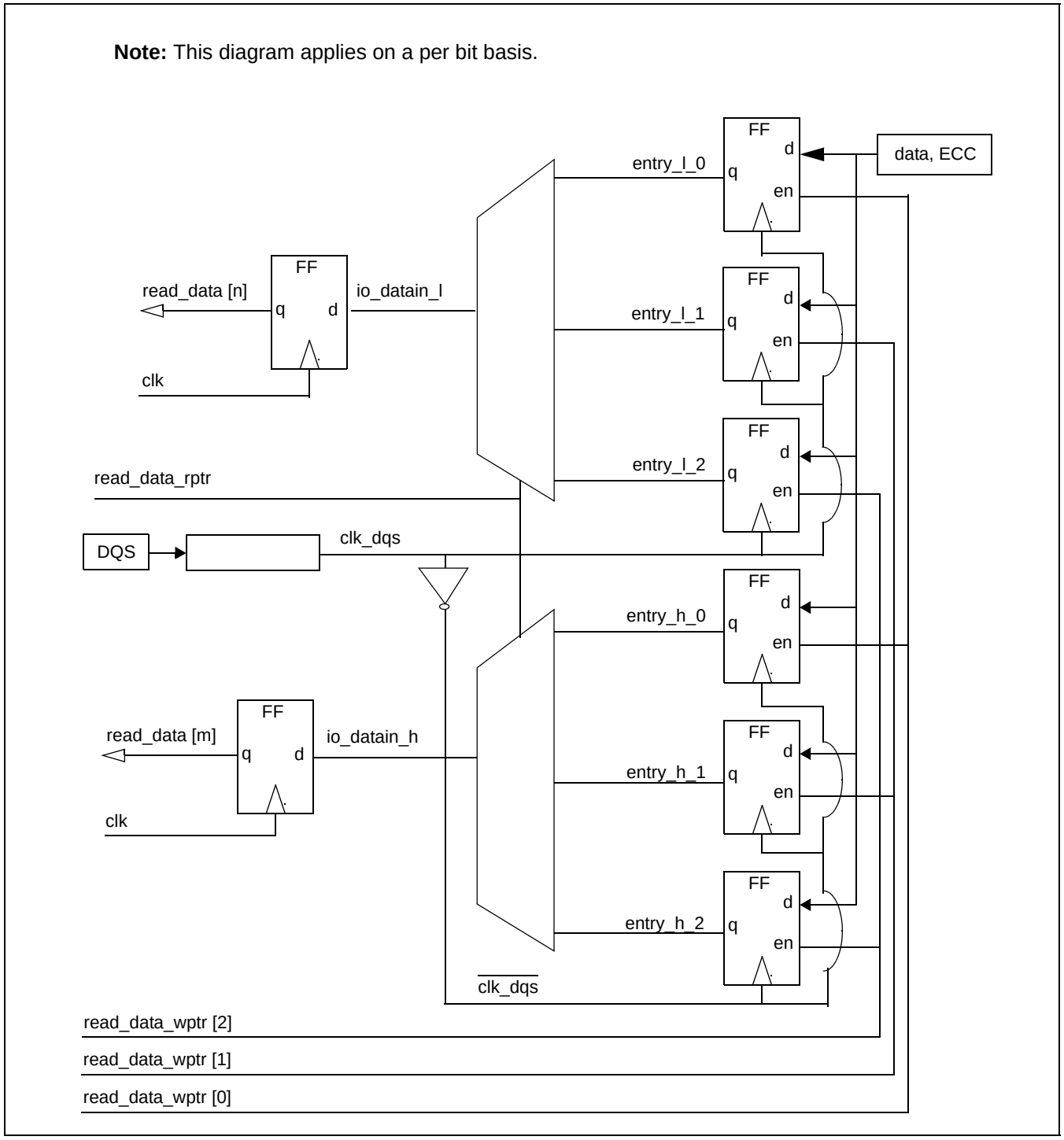
Once the data from the DRAM devices has been captured with the delayed data strobe signal, the data must be transferred to registers that are clocked by the main controller clock. This is done by means of a classic synchronization FIFO as shown in *Figure 19-45*

Each capture register is clocked by the delayed version of DQS (clk_dqs or $\overline{\text{clk\_dqs}}$) and will hold the valid captured data for 3 cycles. This data is then multiplexed and transferred to the main clock domain. The read_data_readptr signals are synchronous to the main clock domain and scheduled so that the entry_h_x and entry_l_x signals are always valid and stable.

Table 19-14. Read Capture Signal Descriptions

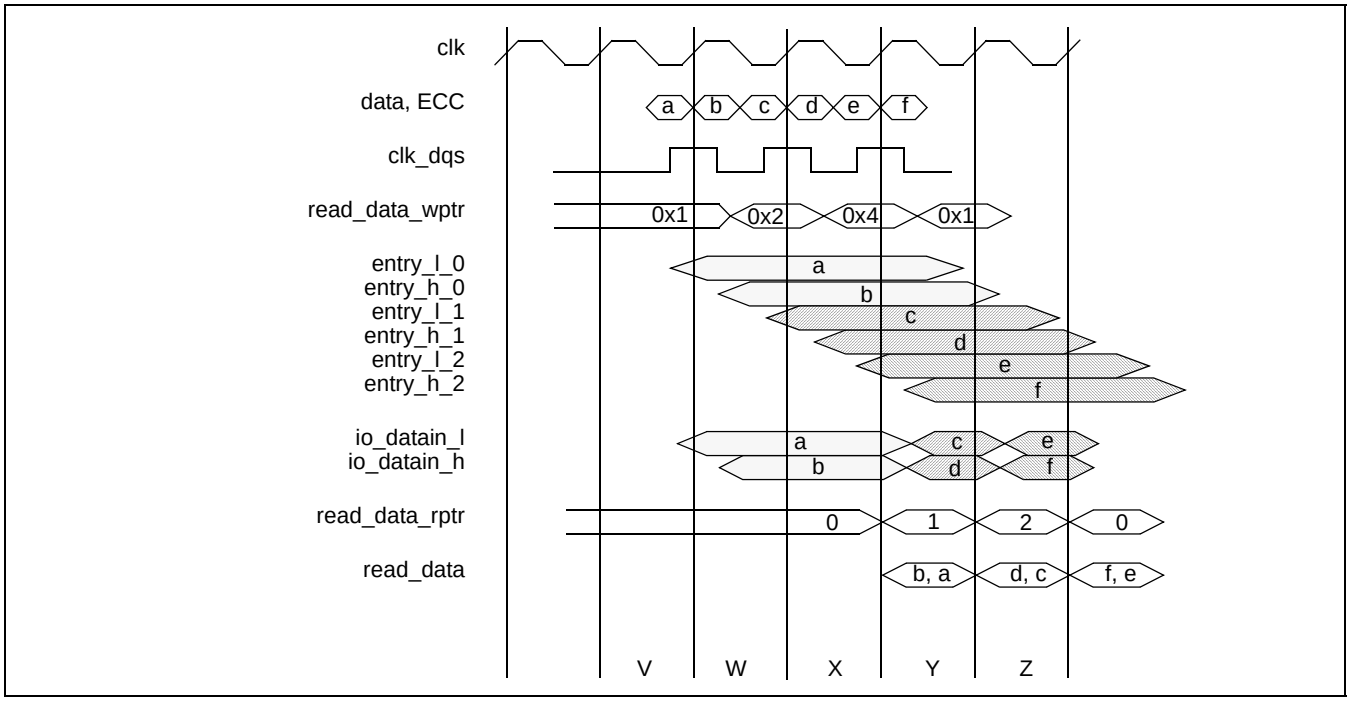
Signal	Type	Description
data, ECC	Chip Input	Incoming data from the DRAM devices.
clk	clock	Main controller clock, synchronous and in phase with DDR_CLK_OUT.
clk_dqs	Clock Strobe	Delayed version of the data strobe from the DRAM devices.
$\overline{\text{clk\_dqs}}$	Clock Strobe	Inverted delayed version of the data strobe from the DRAM devices.
read_data_wptr	internal	Enable signal that controls which set of registers to use when capturing data. This signal is synchronous to the data strobe read domain.
read_data_rptr	internal	Selects which set of registers to choose when transferring data from the data strobe domain to the system clock domain. This signal transitions based on the timing of the main clock.
read_data [m,n]	internal	Synchronized read data from each data signal. n = Data bit for the lower half of the user word. m = Data bit for the corresponding upper half of the user data word. m = memory data path width + n. Example: For a 32-bit memory datapath, input data bit 2 feeds read_data [2] and read_data [34].

Figure 19-45. Data Capture Logic



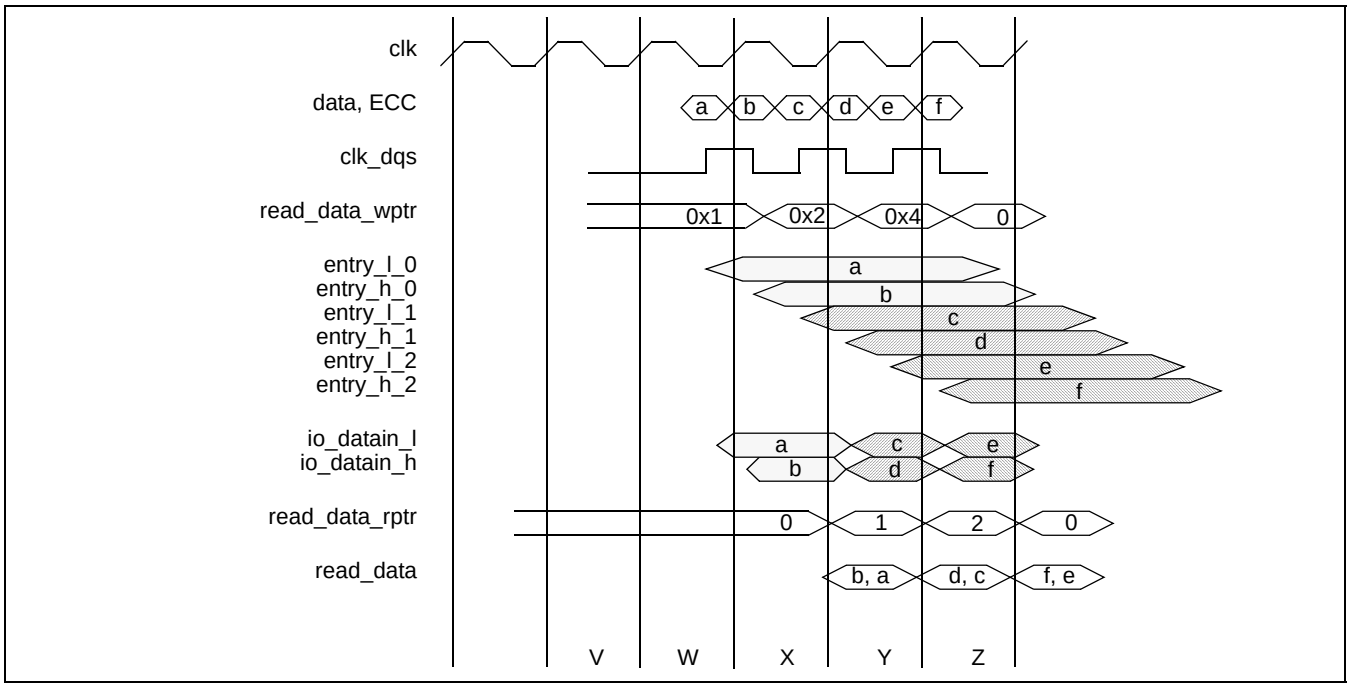
The need to synchronize the data from the DRAM devices to the system clock results from the data and the strobe arrival at any time within a one cycle period. As shown in *Figure 19-46*, the first data word (a) can arrive anywhere from the second half of cycle V to the first half of cycle W. *Figure 19-47* shows the arrival at the latest possible time. In each figure, the data (a/b) is always available to be registered on the rising edge of cycle Y. The next set of data (c/d) is always available to be registered on the rising edge of cycle Z. Data (e/f) are registered on the next rising edge.

Figure 19-46. Read Data Timing (Earliest)



The differences between the earliest and latest arrival times are caused by the large range between minimum and maximum delays of the input cells of the , as well as by the distance from the DRAM devices to the . The memory controller currently is designed to handle all possible delay values up to a maximum of one cycle for the data input pins of the DRAM to the D input of the read capture logic flip-flops.

Figure 19-47. Read Data Timing (Latest)

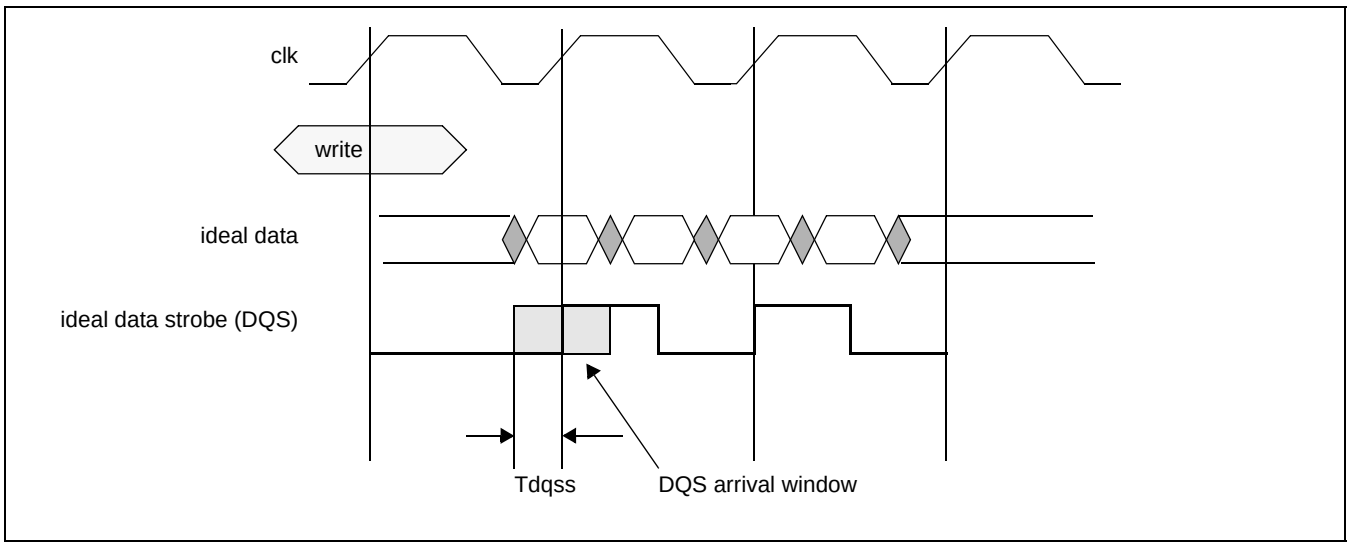


The registers entry_l_x and entry_h_x are loaded every third cycle. The entire path from the DRAM devices to the synchronized data can be considered a multi-cycle path. The diagrams show the timing for whole CAS latencies. For half CAS latencies, the capture of the entry_h/l registers is shifted forward by1/2 clock.

19.8 Write Data Timing

DRAM devices require that the DQS data strobe arrive at the DRAM devices within a certain window around the clock. *Figure 19-48* describes this relationship. The value for Tdqss is specified in fractions of a clock cycle. Most DRAM devices specify this value between ± 0.25 and 0.2 of a clock cycle. This translates to a valid window of between 0.4 and 0.5 of a clock cycle.

Figure 19-48. DRAM DQS Arrival Time Requirements



The data transfer timing from the controller to the DRAM for writes is similar to the read transfer from the DRAM devices to the controller. However, there are two differences:

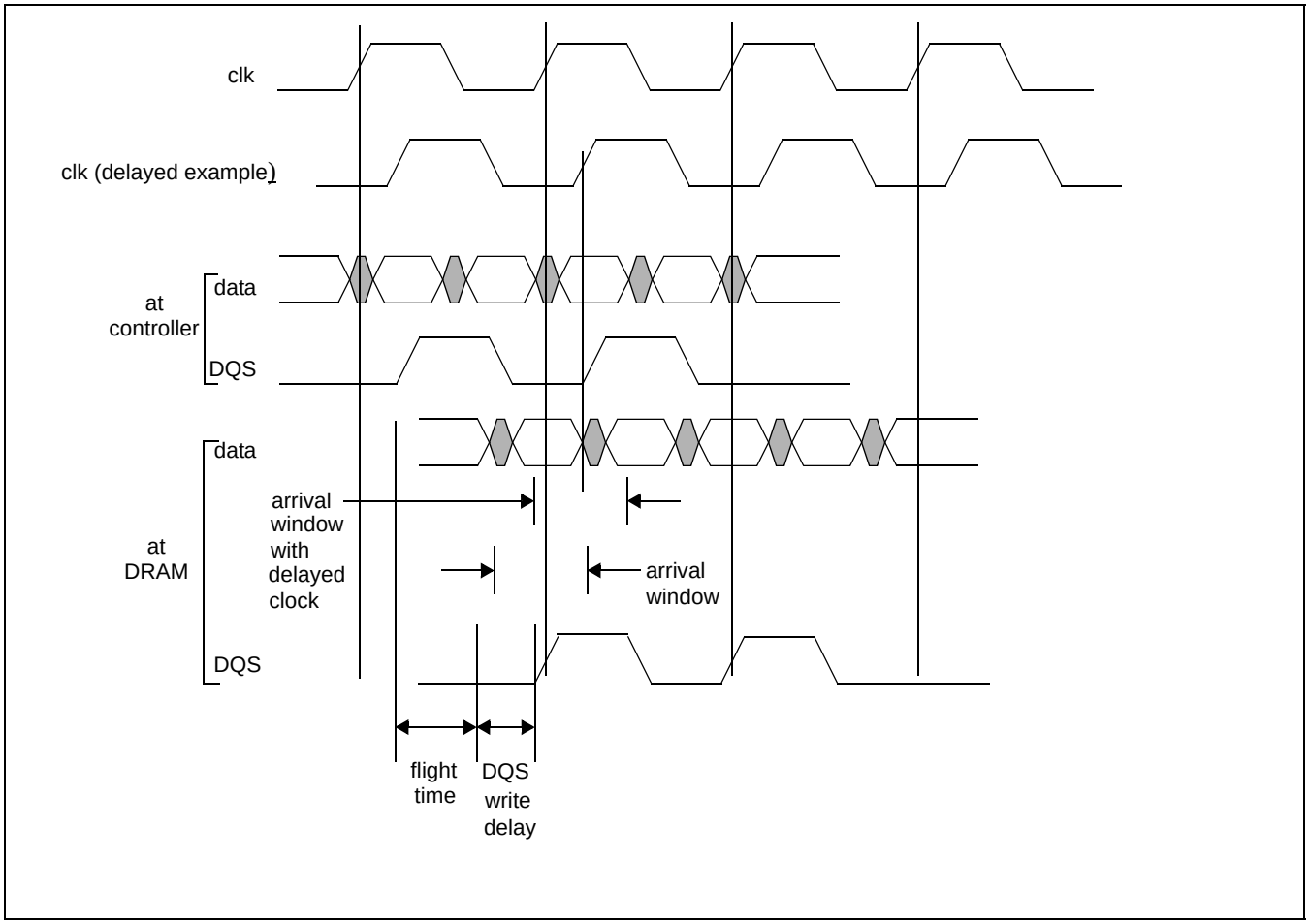
- The DRAM devices expect the data strobe signal to be shifted by the controller to allow the DRAM the maximum margin for capturing the data with the data strobe signal sent to the DRAM devices from the controller.
- The first rising edge of the data strobe signal sent from the controller must occur near the rising edge of the clock at the DRAM. This is called the arrival window. DRAM devices typically specify this window as 0.75clk to 1.25clk for DDR2. Refer to *Figure 19-49* for details.

The DDR interface is source synchronous, which is to say that the sends the clock to the DRAM devices along with the address/control, data and DQS signals. The also maintains two delay lines for controlling the timing of the write data and the write data strobe (DQS). The first delay line adjusts the DQS out transition with respect to the clock edge at the DRAM, and is controlled by the parameter DQS_OUT_SHIFT. The second delay line adjusts the output write data and should be adjusted to maximize the setup and hold requirements around the DQS on writes; it is controlled by the parameter WR_DQS_SHIFT.

Note that in the case of unbufferred DIMMS with heavy loading on the address bus, the main clock to the DRAM devices must be delayed on the board to give sufficient address/command setup time, as detailed in the Datasheet. In this case, the DQS will appear to be shifted "left" or early with respect to the clock at the DRAM devices. In this configuration, it is important that the delay line be set at maximum (1 cycle). The system designer must still make sure that the clock delay inserted does not exceed Tdqss, otherwise the DQS will be too early with

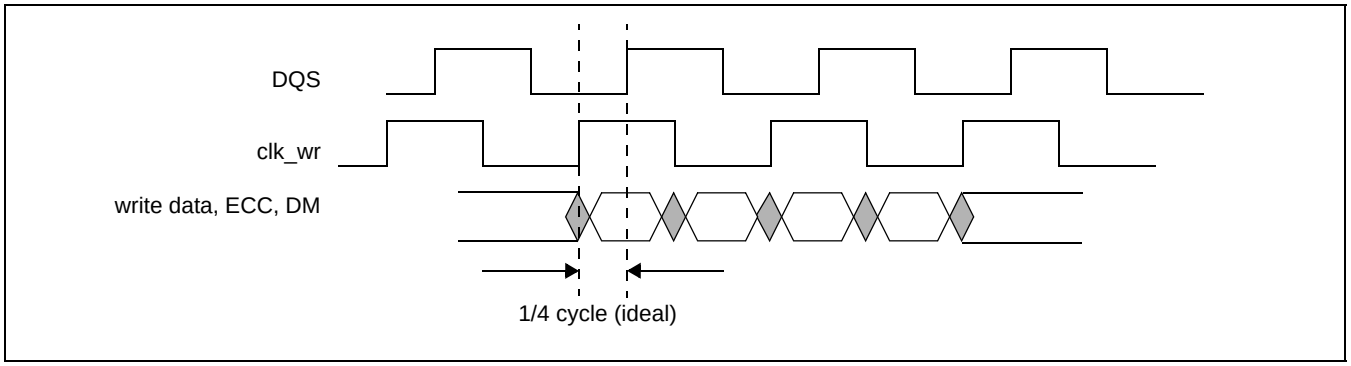
respect to the clock. The following figure shows two cases, one where the clock is not delayed on the board, and the other where it is delayed. Note that in the case of the delayed clock, the DQS and data timing with respect to the do not change, however, because the clock is delayed, they now fall at the left edge of the arrival window (Tdqss minimum).

Figure 19-49. DQS Write Timing



The write data sent along with the data strobe must be aligned such that the strobe rises and falls within the valid region of the data with maximum setup and hold characteristics. This translates into the write data being clocked 1/4 cycle before the rising edge of the data strobe. This relationship is illustrated in *Figure 19-50*, where clk_wr is the name of the signal that clocks the data out of the chip.

Figure 19-50. Write Data and DQS Relationship



The timing of both the clk_wr and DQS signals are controlled by the programmable parameters WR_DQS_SHIFT and DQS_OUT_SHIFT. These parameters allow these two clocks to be delayed a fixed percentage of a cycle from the core clock.

20. Internal SRAM Controller

The Internal SRAM Controller (ISC) transfers data between the on-chip SRAM (OCM) and the Processor Local Bus (PLB). The ISC is the 128-bit interface to the SRAM which is configured as a single 128-bit wide, 16-KB bank (1Kb x 128b).

The ISC supports the following features:

- One bank (Bank 0) of 16KB configurable as 4KB, 8KB or 16KB (128 bits wide)
- 128-bit slave attachment addressable by any PLB master
- Transfers by PLB slave cycles:
  - Single-beat read and write (1 to 8 bytes for 64-bit masters, 1 to 16 bytes for 128-bit masters)
  - 4-word line read and write
  - 8-word line read and write
  - Double word read and write bursts for 64-bit masters
  - Quadword read and write bursts for 128-bit masters
  - Slave-terminated double word and quadword fixed length bursts
  - Master-terminated variable length bursts
- Guarded memory access on 4 KB boundaries
- Data parity checking
- Data transfers occur at PLB bus speeds.
- Power management

20.1 SRAM Controller Registers

The Internal SRAM Controller registers listed in *Table 20-1* are accessed by using move to device control register (**mtdcr**) and move from device control register (**mfdcr**) instructions.

Table 20-1. SRAM Registers

Mnemonic	Name	DCR Address	Access	Page
SRAM0_SB0CR	SRAM Bank 0 Configuration Register	0x380	R/W	448
SRAM0_BEAR	SRAM Bus Error Address Register	0x384	R/W	448
SRAM0_BESR0	Bus Error Status Register 0	0x385	R/W	449
SRAM0_BESR1	Bus Error Status Register 1	0x386	R/W	450
SRAM0_PMEG	Power Management Register	0x387	R/W	452
SRAM0_CID	Core ID Register	0x388	Read	452
SRAM0_REVID	Revision ID Register	0x389	Read	453
SRAM0_DPC	Data Parity Check Register	0x38A	R/W	453

20.1.1 SRAM Bank Configuration Register (SRAM0_SB0CR)

To ensure proper address decode and translation by the SRAM controller, the base address is truncated to the twenty most significant bits and written to SRAM0_SB0CR[0:19]. The default base address is 0xE0010. SB0CR[20:22] is configurable to either 010 for 16KB, 001 for 8KB or 000 for 4KB. The default is 16KB. See *Errors* on page 453 for additional programming notes.

Figure 20-1. Memory Configuration (SRAM0_SB0CR)

0: 19	BAS	Base Address Select	Sets the base address for an SRAM range. The BAS field is compared to bits 0:19 of the effective address. If the effective address is within the range of the starting address plus bank size, the associated bank is enabled for the transaction.
20:22	BS	Bank Size 000 4 KB 001 8 KB 010 16 KB 011 Reserved 100 Reserved 101 Reserved 110 Reserved 110 Reserved	Specifies the size of the logical bank address range. Sets the number of bytes which the bank may access, beginning with the base address set in the BAS field.
23:24	BU	Bank Usage 00 Disabled 01 Bank is valid for read only (RO) 10 Reserved 11 Bank is valid for read/write (R/W)	Protects banks of physical devices from read or write accesses.
25:27		Reserved	
28:31	UA	Four least significant bits of the upper 32-bit address.	Must be 0b0000.

When an attempt is made to write access to an address within the range of the BAS field and the bank is designated as read-only, an SRAM controller protection error occurs.

20.1.2 Bus Error Address Register (SRAM0_BEAR)

The Bus Error Address Register (SRAM0_BEAR) is a 32-bit register which contains the address of the access where a data bus error has occurred. The BEAR is written when a data access error occurs and its contents may be locked until the BEAR lock bit in the Bus Error Status Register 0 (BESR0) is cleared, depending on the state of the LOCK ERROR signal when the transaction was accepted. The contents of the BEAR can be accessed using the move from device control register (**mfdcr**) and move to device control register (**mtdcr**) instructions.

Figure 20-2. Bus Error Address Register (SRAM0_BEAR)

0:31	ABE	Address of Bus Error (asynchronous)	

20.1.3 Bus Error Status Register 0 (SRAM0_BESR0)

The Bus Error Status Register 0 (SRAM0_BESR0) records the occurrence and type of errors for transactions attempted on behalf of each PLB master. The contents of the SRAM0_BESR0 can be accessed using the move from device control register (**mfdcr**) and move to device control register (**mtdcr**) instructions.

The field lock bit protects fields ET0, RWS1, FL2, and AL3 for master 0, 1, 2, and 3. Upon error detection, any of master n fields may be overwritten if the field lock bit for that master is zero. If the field lock bit for a particular master has been set to zero by the DCR master, the BEAR lock bit for that master will be overwritten when the next PLB error is detected, regardless of the status of the BEAR lock prior to the error. If the BEAR lock is overwritten as a zero, the BEAR will be overwritten on the next error detected from that master, but will not be overwritten if the BEAR lock is due to an error caused by another master.

Figure 20-3. Bus Error Status Register 0 (SRAM0_BESR0)

0:2	ET0	Error type for master 0 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 0 is processor core instruction cache unit
3	RWS0	Read/write status for master 0 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL0	Field lock for master 0 0 Fields are unlocked 1 Fields are locked	
5	AL0	BEAR address lock for master 0 0 BEAR address unlocked 1 BEAR address locked	
6:8	ET1	Error type for master 1 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 1 is processor core data cache read unit
9	RWS1	Read/write status for master 1 0 Error operation was a write operation 1 Error operation was a read operation	
10	FL1	Field lock for master 1 0 Fields are unlocked 1 Fields are locked	
11	AL1	BEAR address lock for master 1 0 BEAR address unlocked 1 BEAR address locked	

12:14	ET2	Error type for master 2 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 2 is processor core data cache write unit
15	RWS2	Read/write status for master 2 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL2	Field lock for master 2 0 Fields are unlocked 1 Fields are locked	
17	AL2	BEAR address lock for master 2 0 BEAR address unlocked 1 BEAR address locked	
18:20	ET3	Error type for master 3 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 3 is the Direct Memory Access (DMA2P40) controller
21	RWS3	Read/write status for master 3 0 Error operation was a write operation 1 Error operation was a read operation	
22	FL3	Field lock for master 3 0 Fields are unlocked 1 Fields are locked	
23	AL3	BEAR address lock for master 3 0 BEAR address unlocked 1 BEAR address locked	
24:31		Reserved	

20.1.4 Bus Error Status Register 1 (SRAM0_BESR1)

The Bus Error Status Register 1 (SRAM0_BESR1) records the occurrence and type of errors for transactions attempted on behalf of each PLB master. The contents of the SRAM0_BESR1 can be accessed using the move from device control register (mfdcr) and move to device control register (mtdcr) instructions.

The field lock bit protects fields ET4, RWS5, FL6, and AL7 for masters 4, 5, 6, and 7. Upon error detection, any of master n fields may be overwritten if the field lock bit for that master is zero. If the field lock bit for a particular master has been set to zero by the DCR master, the BEAR lock bit for that master will be overwritten when the next PLB error is detected, regardless of the status of the BEAR lock prior to the error. If the BEAR lock is overwritten as a zero, the BEAR will be overwritten on the next error detected from that master, but will not be overwritten if the BEAR lock is due to an error caused by another master.

Figure 20-4. Bus Error Status Register 1 (SRAM0_BESR1)			
0:2	ET4	Error type for master 4 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 4 is PLB3 to PLB4 bridge
3	RWS4	Read/write status for master 4 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL4	Field lock for master 4 0 Fields are unlocked 1 Fields are locked	
5	AL4	BEAR address lock for master4 0 BEAR address unlocked 1 BEAR address locked	
6:8	ET5	Error type for master 5 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 5 is OPB to PLB4 bridge
9	RWS5	Read/write status for master 5 0 Error operation was a write operation 1 Error operation was a read operation	
10	FL5	Field lock for master 5 0 Fields are unlocked 1 Fields are locked	
11	AL5	BEAR address lock for master 5 0 BEAR address unlocked 1 BEAR address locked	
12:14	ET6	Error type for master 6 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 6 is media access layer
15	RWS6	Read/write status for master 6 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL6	Field lock for master 6 0 Fields are unlocked 1 Fields are locked	
17	AL6	BEAR address lock for master 6 0 BEAR address unlocked 1 BEAR address locked	

18:20	ET7	Error type for master 7 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 7 is the Security Engine
21	RWS7	Read/write status for master 7 0 Error operation was a write operation 1 Error operation was a read operation	
22	FL7	Field lock for master 7 0 Fields are unlocked 1 Fields are locked	
23	AL7	BEAR address lock for master 7 0 BEAR address unlocked 1 BEAR address locked	
24:31		Reserved	

20.1.5 Power Management Register (SRAM0_PMEG)

The Power Management Register (SRAM0_PMEG) enables the sleep function for the SRAM controller.

Figure 20-5. Power Management Register (SRAM0_PMEG)			
0	PMEN	Power Management enable: 0 Sleep mode disabled 1 Sleep mode enabled	
1:6	PMCNT	Power Management Counter	The value (n) programmed into these register bits is a multiple of 16 clock cycles (n x 16). If PM_EN is set to 1, the SRAM controller goes into sleep mode after being idle for n x 16 clock cycles.
7:10	PMDFLT	Power Management Default Wait Interval Hard wired to 1111.	This field is read-only. The hard wired value is 16 clock cycles. If PM_EN is set to 1, SRAM controller goes into sleep mode after being idle for 16 clock cycles.
11:31		Reserved	

20.1.6 Core ID Register (SRAM0_CID)

The read-only Core ID Register (SRAM0_CID) identifies the core number for the SRAM controller core.

Figure 20-6. SRAM Internal Core Device ID Register (SRAM0_CID)			
0:31	CID	Internal Core Device ID	H322B0000

20.1.7 Revision ID Register (SRAM0_REVID)

The read-only Revision ID Register (SRAM0_REVID) identifies the project number, SCCS version, and netlist number for the SRAM controller core.

Figure 20-7. SRAM Revision ID Register (SRAM0_REVID)			
0:7	PROJ	Project Number	
8:23	SCCS	SCCS Version	
24:31	NLST	Netlist Number	

20.1.8 Data Parity Checking Register (SRAM0_DPC)

The Data Parity Checking register (SRAM0_DPC) enables data parity generation and checking if the data parity function is implemented. It controls data parity generation during write transactions and data parity checking during read transactions.

Note: After a core reset, the default DPC value is 1.

Figure 20-8. Data Parity Checking Register (SRAM0_DPC)			
0	DPC	Data Parity Check: 0 Disabled 1 Enabled	When set to 1, this bit enables data parity generation during write transactions and read transactions. When set to 0, this bit disables both functions.
1:31		Reserved	

20.2 Errors

The SRAM controller monitors two types of errors when executing PLB transfers: Protection and Data Parity errors.

20.2.1 Protection Error

A Protection error occurs when the requested read or write operation violates the bank usage programming, for example, when a write is attempted on read-only bank.

When the SRAM detects an error, it reports this error condition to the owning master using a unique ERROR line. When a master requests a transfer, it also provides a unique master ID number that is encoded in the PLB_masterID. This value is latched in a register and when an error occurs during this master's tenure, then it owns the error and, the internal SRAM controller drives the corresponding ERROR signal back to this master. The internal SRAM controller logs the type of error into the appropriate BESR and the address at which this error occurred into the BEAR. There is one error field for each master, but only one BEAR.

20.2.1.1 BESR Field

If a master does not assert the transfer qualifier LOCK ERROR, then any error occurring for this master overwrites any previously logged error in the BESR field for this master. The BEAR always has the address of the most recently detected error if lockerror has not been asserted and no master has its own field address bit locked.

20.2.2 Data Parity Error

A Data Parity error occurs when the parity of data during a read transaction does not match the parity that is generated and stored for the specific data. Generation of stored parity occurs during write transactions. The operation of parity error is identical to protect error. The SRAM0_BEAR contains the address at which parity error occurred. Unlike with the protect error, the address captured in the SRAM0_BEAR for parity error could be one address beyond where the parity error occurred. SRAM0_BESRx fields also log parity errors for the particular master, depending on the state of the lock error signal.

21. Peripheral Component Interconnect (PCI) Interface

The peripheral component interconnect (PCI) interface and bridge (referred to asthe PCI bridge in this chapter) provides a means for connecting PCI-compatible devices to the on-chip bus architecture of the PPC440EPx/GRx chip. The PCI bridge is designed to the *PCI Specification*, Version 2.2. The PCI bridge is bidirectional in that it allows PPC440EPx/GRx PLB masters to access PCI targets off-chip. It also allows PCI masters to access PLB slave devices such as the SDRAM controller. The PCI bridge contains an arbiter that can optionally be used for host applications.

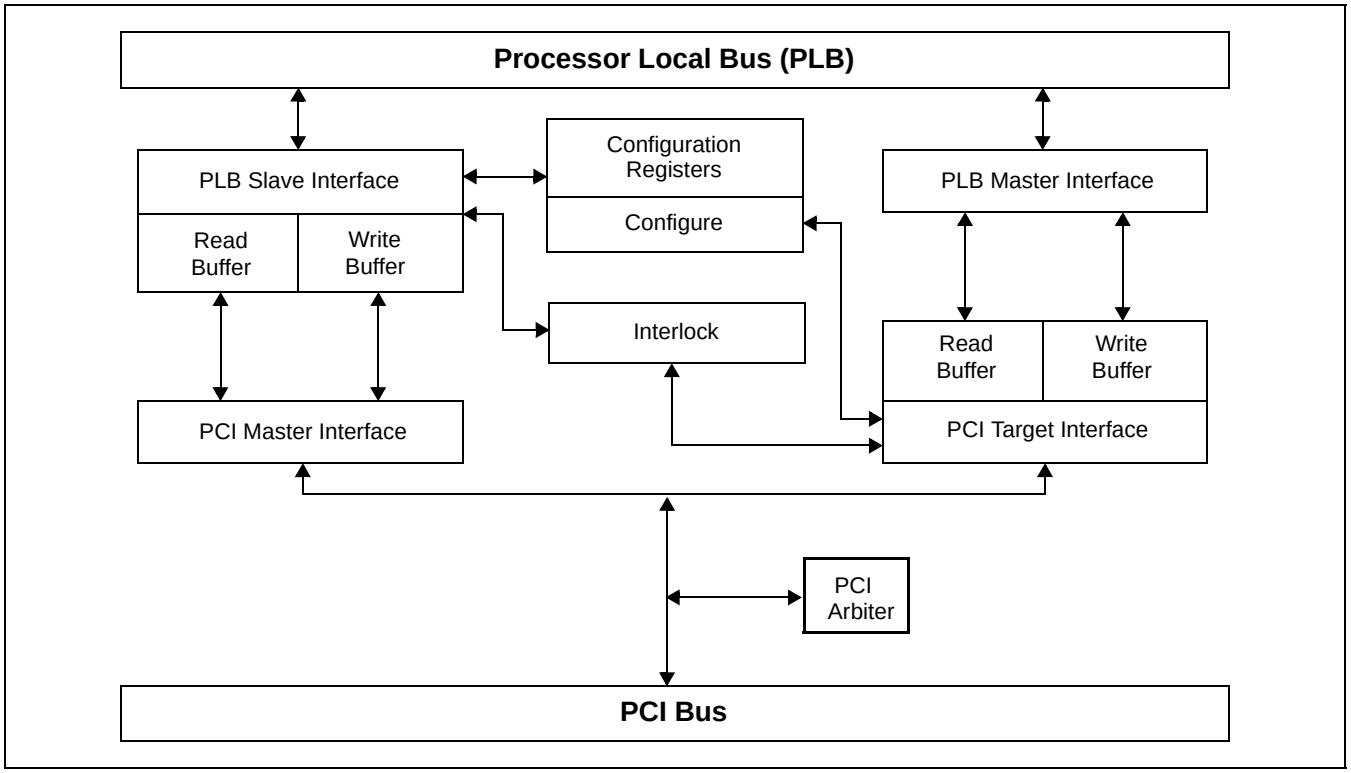
The PCI bridge can be used as the host bridge. The PCI bridge is also configurable by an external PCI agent, allowing it to be used in target adapter applications. The PCI bridge contains address mapping register sets to provide address mapping for both transaction directions (see *Figure 21-1 PCI Bridge Block Diagram*). Agents on the PLB are referred to as masters or slaves. Agents on the PCI are referred to as targets or masters.

21.1 Features

- PCI bus frequency up to 66 MHz (asynchronous)
- Asynchronous clocking between PLB and PCI buses
- Supports 1:1, 2:1, 3:1, and 4:1 clock ratios from PLB to PCI
- 32-bit PCI Address/Data Bus
- Power Management
- Buffering:
  - PCI target 64-byte write post buffer
  - PCI target 96-byte read prefetch buffer
  - PLB slave 32-byte write post buffer
  - PLB slave 64-byte read prefetch buffer
- Error tracking/status
- PCI arbitration function (optional)
- Supports PCI target-side configuration
- Supports processor access to all PCI address spaces:
  - Single-beat PCI I/O reads and writes
  - PCI memory single-beat and prefetch-burst reads and single-beat writes
  - Single-beat PCI configuration reads and writes (type 0 and type 1)
  - PCI interrupt acknowledge

Figure 21-1 shows the PCI bridge block diagram.

Figure 21-1. PCI Bridge Block Diagram



21.2 Byte Ordering

The PCI bridge configuration register address space must be treated as little endian, as required by *PCI Specification*, Version 2.2. In most cases data memory areas in PCI address space will be configured and used in little endian format. To provide for this, PCI configuration space and memory map regions should be defined as little endian memory space by means of the corresponding entry in the 440EPx CPU's MMU.

Byte ordering and management of little endian memory space from a PowerPC CPU point of view is described in detail in *Byte Ordering* in the *PPC440 Processor User's Manual*. PowerPC architecture and CoreConnect bus architecture both use a bit naming convention in which the most significant bit (msb) name incorporates the numeral 0 and the least significant bit (lsb name for a 32-bit vector incorporates the numeral 31. *Table 21-1* shows the correspondence of address bit-naming conventions for PowerPC, CoreConnect PLB, and PCI interface.

Table 21-1. PowerPC, CoreConnect PLB, and PCI Address Bit-Naming Conventions

Functional Unit/Interface	Word Address	Byte Address
PPC440EPx/GRx Processor Core Address	A0:29	A30:31
CoreConnect — PLB Address Bus	PLB_ABus0:29	PLB_ABus30:31
PCI Address Bus	AD31:2	AD1:0

Table 21-2 shows the correspondence of data bus bit naming conventions and data lane connections for PowerPC, CoreConnect PLB, and PCI interface. Note that within a data lane (column), the data signal naming indicates that, for example, AD31is connected to PLB Write Data24.

Table 21-2. PowerPC, CoreConnect PLB, and PCI Data Bus Bit-Naming Conventions

Functional Unit/ Interface	Most Significant Byte (MSB)	<–>	<–>	Least Significant Byte (LSB)
Data Byte Value (0x*nn*)	11	22	33	44
Little Endian Byte Address (0b*nn*)	11	10	01	00
PPC440EPx/GRx Processor Core (Write) Data Bus	Data24:31	Data16:23	Data8:15	Data0:7
CoreConnect — PLB Write Data Bus — Byte Group	PLB Write Data24:31	PLB Write Data16:23	PLB Write Data8:15	PLB Write Data0:7
PLB Byte Enable	PLB_BE3	PLB_BE2	PLB_BE1	PLB_BE0
PCI Byte Enable	C/BE3	C/BE2	C/BE1	C/BE0
PCI Data Bus — Byte Group	AD31:24	AD23:16	AD15:8	AD7:0

1. Logical data work (32-bit word) == 0x11223344
2. 440 CPU performing either:
- Store word to little endian memory space
- Store word—byte reversed—to big endian address space

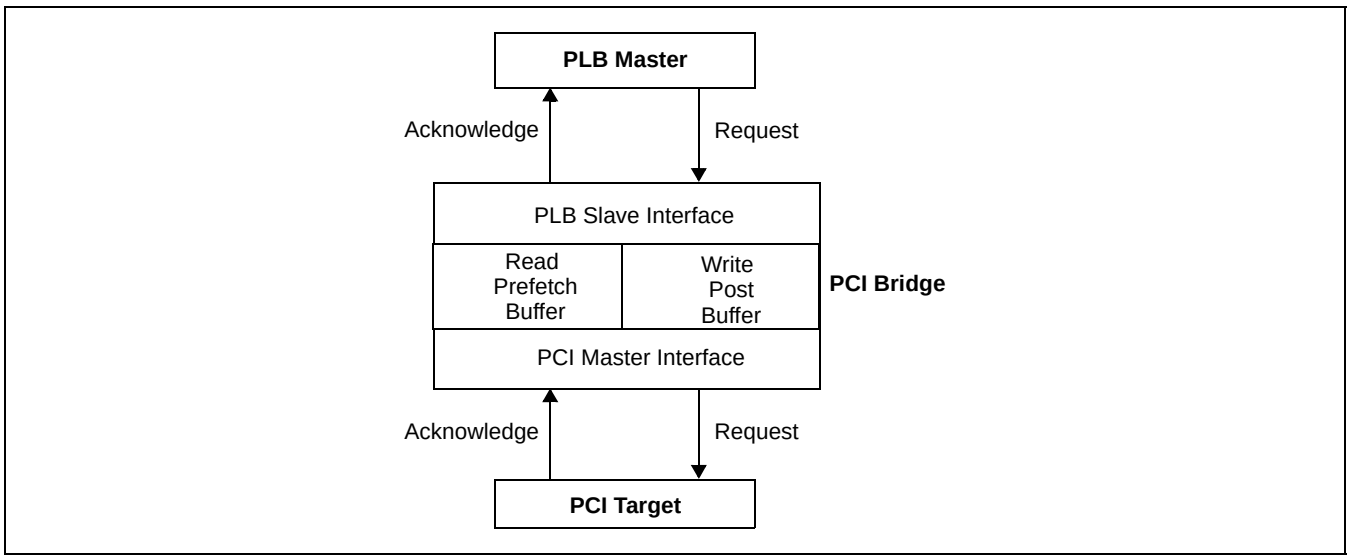
21.3 PCI Bridge Functional Blocks

The following sections describe the PCI bridges and the associated arbiter.

21.3.1 PLB-to-PCI Half-Bridge

As shown in *Figure 21-2*, the 64-bit PLB slave interface and PCI master interface function together as a PLB-to-PCI half-bridge to enable PLB master devices to access PCI target devices. The half-bridge configuration contains a 32-byte write post buffer and a 64-byte read prefetch buffer.

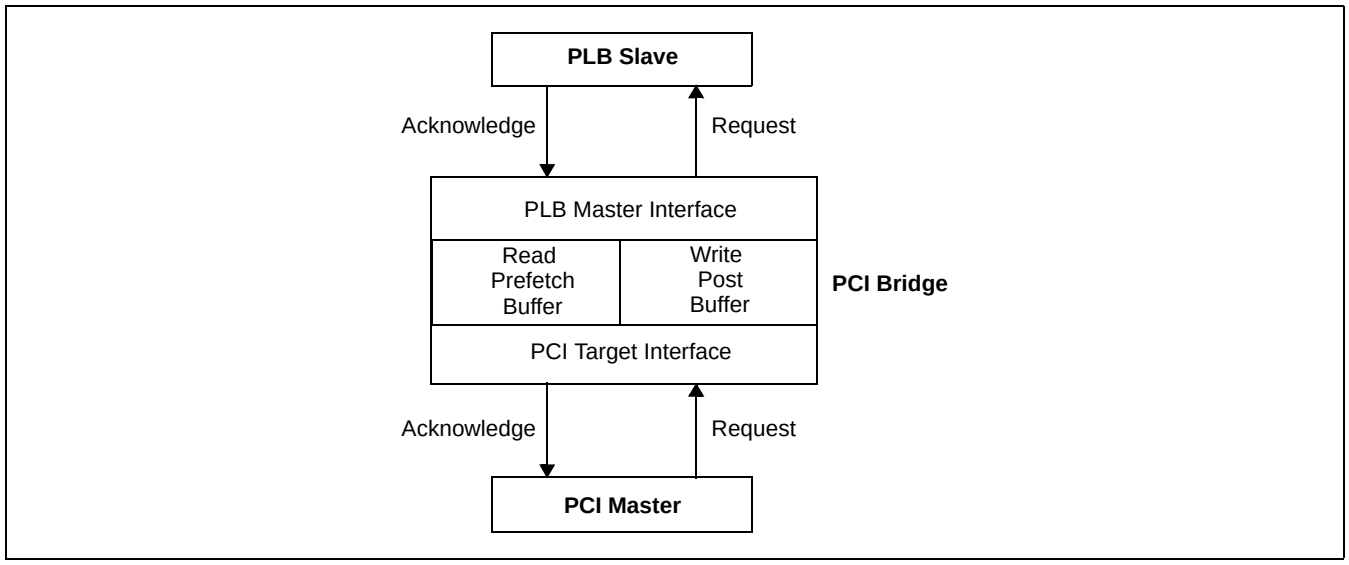
Figure 21-2. PLB-to-PCI Half-Bridge Block Diagram



21.3.2 PCI-to-PLB Half-Bridge

As shown in *Figure 21-3*, the PCI target interface and 64-bit PLB master interface function together as a PCI-to-PLB half-bridge to enable PCI master devices to access PLB slave devices. The half-bridge configuration contains a 64-byte write post buffer and a 96-byte read prefetch buffer.

Figure 21-3. PCI-to-PLB Half-Bridge Block Diagram

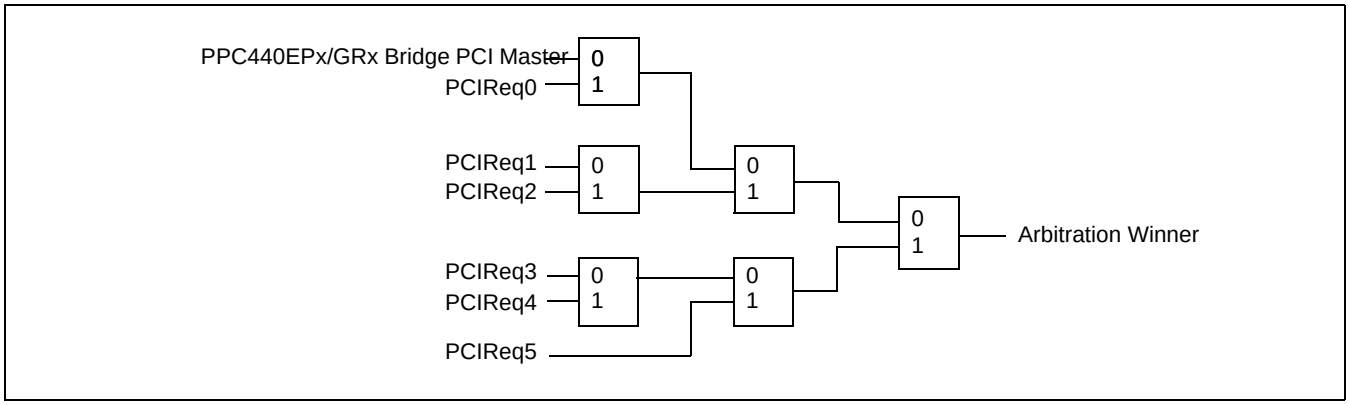


21.3.3 PCI Arbiter

The internal arbiter can be used with up to six external PCI masters ($\overline{Req}$/$\overline{Gnt}$ pairs) or can be disabled. When the internal arbiter is disabled, there is one $\overline{Req}$/$\overline{Gnt}$ pair that must be attached to an external arbiter. A strapping configuration pin determines whether the internal arbiter is enabled or not. Priority is round-robin (rotating). Priority switches when a master begins a transfer by asserting $\overline{Frame}$. Each block keeps a priority bit that only switches if its highest priority requestor receives a grant. Assuming that all priority bits are initially cleared and all requests are active, an example rotation would be 440EPx 2, 1, 4. *PCI Specification*, Version 2.2 requires that all PCI devices three-state their pins during reset. The 440EPx PCI arbiter supports bus parking during normal operation.

Figure 21-4 shows the logical arbitration structure.

Figure 21-4. Arbitration Structure



21.4 PCI Bridge Address Mapping

The following sections describe the address maps supported by the PCI bridge.

21.4.1 PLB-to-PCI Address Mapping

The PCI bridge responds as a slave on the PLB bus in several address ranges. These ranges enable a PLB master to configure the PCI bridge, and to cause the PCI bridge to generate memory, I/O, configuration, interrupt acknowledge, and special cycles to the PCI bus. *Table 21-3* shows the address map from the view of the PLB, that is, as decoded by the PCI bridge as a PLB slave.

Table 21-3. PLB Address Map

PLB Address Range	Description	PCI Address Range
0x1 E800 0000–0x1 E800 FFFF	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 0 to 64KB − 1.	0x00000000–0x0000FFFF
0x1 E801 0000–0x1 E87F FFFF	Reserved PCI bridge does not respond. (Other bridges use this space for non-contiguous I/O.)	
0x1 E880 0000–0x1 EBFF FFFF	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 8MB to 64MB − 1.	0x00800000–0x03FFFFFF
0x1 EC00 0000–0x1 EEBF FFFF	Reserved PCI bridge does not respond	
0x1 EEC0 0000–0x1 EECF FFFF	PCIC0_CFGADDR and PCIC0_CFGDATA 0x1 EEC00000: PCIC0_CFGADDR 0x1 EEC00004: PCIC0_CFGDATA 0x1 EEC00008–0x1 EECFFFFF: Reserved (can mirror PCIC0_CFGADDR and PCIC0_CFGDATA).	
0x1 EED0 0000–0x1 EEDF FFFF	PCI Interrupt Acknowledge and Special Cycle 0x1 EED00000 read: Interrupt Acknowledge 0x1 EED00000 write: Special Cycle 0x1 EED00004–0x1 EEDFFFFF: Reserved (can mirror Interrupt Acknowledge and Special Cycle).	
0x1 EEE0 0000–0x1 EF3F FFFF	Reserved PCI bridge does not respond.	

Table 21-3. PLB Address Map (continued)

PLB Address Range	Description	PCI Address Range
0x1 EF40 0000–0x1 EF4F FFFF	PCI Bridge Local Configuration Registers 0x1 EF400000: PCIL0_PMM0LA 0x1 EF400004: PCIL0_PMM0MA 0x1 EF400008: PCIL0_PMM0PCILA 0x1 EF40000C: PCIL0_PMM0PCIHA 0x1 EF400010: PCIL0_PMM1LA 0x1 EF400014: PCIL0_PMM1MA 0x1 EF400018: PCIL0_PMM1PCILA 0x1 EF40001C: PCIL0_PMM1PCIHA 0x1 EF400020: PCIL0_PMM2LA 0x1 EF400024: PCIL0_PMM2MA 0x1 EF400028: PCIL0_PMM2PCILA 0x1 EF40002C: PCIL0_PMM2PCIHA 0x1 EF400030: PCIL0_PTM1MS 0x1 EF400034: PCIL0_PTM1LA 0x1 EF400038: PCIL0_PTM2MS 0x1 EF40003C: PCIL0_PTM2LA 0x1 F400040–0x1 EF4FFFFF: Reserved (can mirror PCI local registers)	
0x1 8000 0000–0x1 BFFF FFFF (see **Note**)	PCI Memory—Range 0 PMM 0 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	0x0000 0000 0000 0000– 0xFFFF FFFF FFFF FFFF
0x1 8000 0000–0x1 BFFF FFFF	PCI Memory—Range 1 PMM 1 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	0x0000 0000 0000 0000– 0xFFFF FFFF FFFF FFFF
0x1 8000 0000–0x1 BFFF FFFF	PCI Memory—Range 2 PMM 2 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	0x0000 0000 0000 0000– 0xFFFF FFFF FFFF FFFF
Note: Memory map ranges are fully programmable. The ranges must not overlap with each other or conflict with any other memory mappings.		

Three PCI bridge address ranges, associated with PLB masters in PLB space, are mapped to PCI memory space: PCI master map (PMM) 0, PMM1, and PMM2.

Each PMM is configured using the following registers (*n* is 0, 1, or 2, corresponding with PMM0, PMM1, and PMM2, respectively):

- PMMnLocal Address (PCIL0_PMMnLA)
- PMMnMask/Attribute (PCIL0_PMMnMA)
- PMMnPCI Low Address (PCIL0_PMMnPCILA)
- PMMnPCI High Address (PCIL0_PMMnPCIHA)

The location of each PMM in PLB space is programmable, using the PCIL0_PMMnLA registers. The PLB address range assigned to each PMM should not overlap any other PLB address space range that is used or reserved. Overlapping results in undefined behavior.

The range of PCI memory address space associated with each PMM is also programmable, and is a 64-bit address space to enable address translation between the PCI bus and the PLB. The PCIL0_PMMnPCILA registers contain the low-order word of a PCI address; the PCIL0_PMMnPCIHA registers contain the high-order word of a PCI address. If the high-order word of a PCI address is greater than 0, the PCI bridge generates dual address cycles to the PCI.

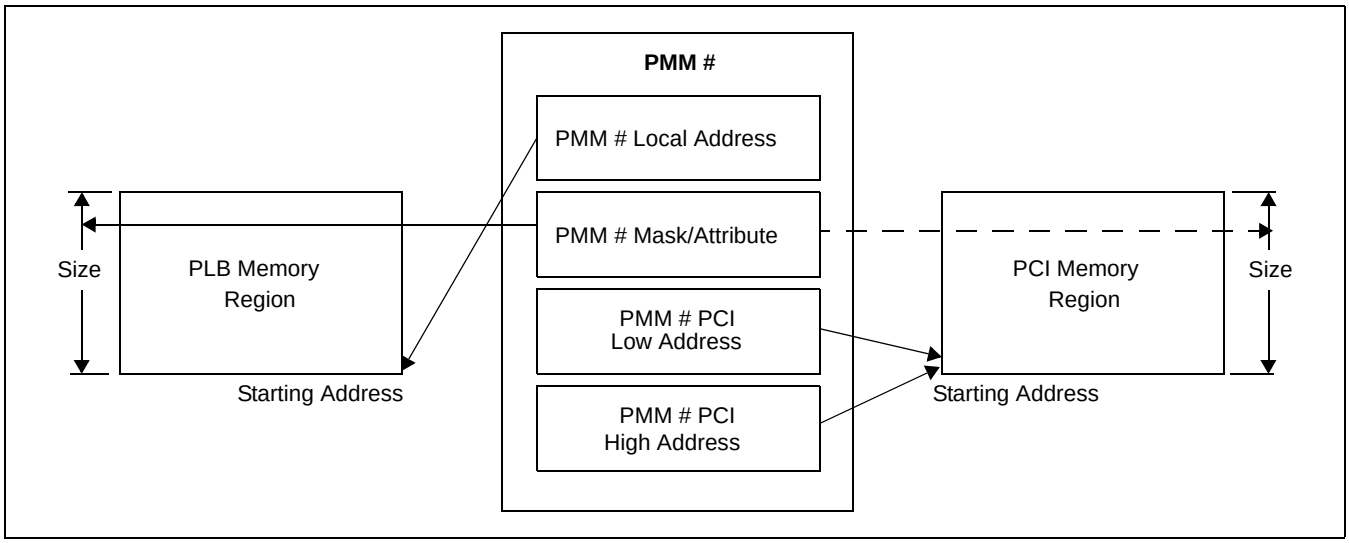
The size of each PMM is programmable, using the mask portion of the PCIL0_PMMnMA registers. The size is a power of 2, ranging from 4KB–4GB. The PLB and PCI address spaces for each PMM are aligned to the size contained in the associated PCIL0_PMMnMA registers.

The attribute portion of the PCIL0_PMMnMA registers specify whether the associated PMM is enabled or disabled, and marked as prefetchable or not prefetchable.

Address ranges and attributes should be initialized before a PMM is enabled.

Figure 21-5 shows the detail of the PMM register sets used to map PLB memory regions to PCI address space.

Figure 21-5. PMM Register Sets Map PLB Address Space to PCI Address Space



21.4.2 PCI-to-PLB Address Mapping

The PCI bridge responds as a PCI target for memory accesses and configuration Type 0 accesses. *Table 21-4* shows the PCI memory address map from the view of PCI, that is, as decoded by the PCI bridge as a PCI target.

Table 21-4. PCI Memory Address Map

PCI Memory Address	Description	PLB Address
0x00000000–0xFFFFFFFF	System Memory or ROM—Range 0 PTM 1 maps a region of PCI memory space to PLB space, which can map to system memory or ROM. Size and location is programmable. The space supports address translation between the PCI and the PLB.	0x00000000–0xFFFFFFFF
0x00000000–0xFFFFFFFF	System Memory or ROM—Range 1 PTM 2 maps a region of PCI memory space to PLB space, which can map to system memory or ROM. Size and location is programmable. The space supports address translation between the PCI and the PLB.	0x00000000–0xFFFFFFFF

21.4.3 PCI Target Map Configuration

Two PCI bridge address ranges in PCI memory space are mapped to PLB space: PCI target map (PTM1) and PTM2 (PTM0 is reserved).

Each PTM is configured using the following registers (n is 1 or 2, corresponding with PTM1 and PTM2, respectively).

- PTMnMemory Size (PCIL0_PTMnMS)
- PTMnLocal Address (PCIL0_PTMnLA)
- PTMnBAR (PCIC0_PTMnBAR)

The size of each PTM is programmable, using the PCIL0_PTMnMS registers. The size is a power of 2, and ranges from 4KB–4GB. The PLB and PCI address spaces for each PTM are aligned to this size.

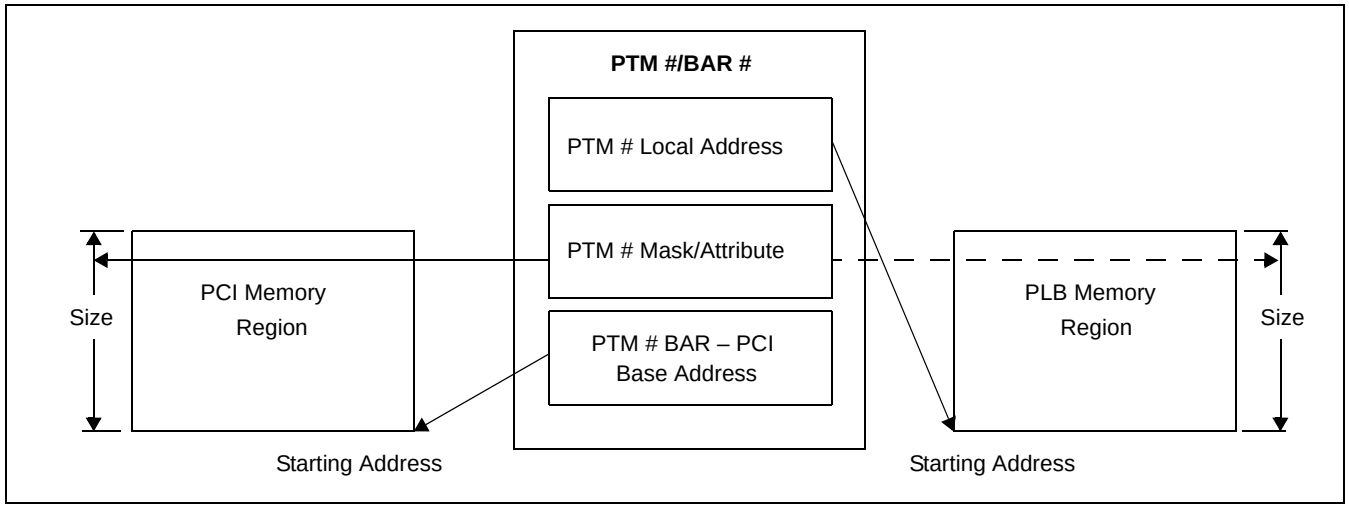
The address range of PLB space accessed through each PTM is also programmable, enabling address translation between the PCI bus and the PLB. The PLB address range is defined in the PCIL0_PTMnLA registers.

The location of each PTM in PCI memory space is programmable, using the PCIC0_PTMnBAR registers.

The PTMs are enabled and disabled using PCIC0_CMD[MA]. PTM address ranges and sizes should be initialized before being enabled. If the PCI bridge is not the host bridge, the local processor must initialize the PTM size before enabling host configuration setting the Host Configuration Enable (HCE) field of the Bridge Options 2 register (PCIC0_BRDGOPT2). This ensures that the host experiences proper behavior from the PCIL0_PTMnBAR registers. Note that PTM1 is always enabled. The PTM1 registers must always be initialized.

Figure 21-6 shows the detail of the PMM/BAR register sets used to map PCI memory regions to PLB address space.

Figure 21-6. PTM Register Sets Map PCI Address Space to PLB Address Space



21.5 PCI Bridge Transaction Handling

The following sections discuss PCI bridge transactions and completion ordering.

21.5.1 PLB-to-PCI Transaction Handling

This section describes how the PCI bridge responds to read and write requests from a PLB master. The PCI bridge decodes and accepts PLB transactions to different address ranges resulting in the generation of memory, I/O, configuration, interrupt acknowledge and special cycles on the PCI bus.

Table 21-5. Transaction Mapping: PLB —> PCI

PLB Transaction PLB Master → Bridge (PLB Slave Interface)	Bridge Mapping and Qualifications	PCI Transaction Bridge (PCI Master Interface) → PCI Target
Single-beat 1 → 8-byte Read	64KB or 56MB PCI I/O address range	I/O Read
Single-beat 1 → 8-byte Write	64KB or 56MB PCI I/O address range	I/O Write
Single-Beat 1 → 8-byte Read	Access to PCIC0_CFGDATA register	Configuration Read (Type 0, 1)
Single-Beat 1 → 8-byte Write	Access to PCIC0_CFGDATA register	Configuration Write (Type 0, 1)
Single-Beat 1 → 8-byte Read	PLB address decodes to PMM0, PMM1, or PMM2, nonprefetchable	Memory Read
Burst Read	PLB address decodes to PMM0, PMM1, or PMM2, nonprefetchable	Memory Read
PLB 4-word and 8-word Line Reads	PLB address decodes to PMM0, PMM1, or PMM2	Memory Read Line
Single-Beat 1 → 4-byte Read	PLB address decodes to PMM0, PMM1, or PMM2, prefetchable	Memory Read Multiple
Burst Read	PLB address decodes to PMM0, PMM1, or PMM2, prefetchable	Memory Read Multiple
Single-Beat 1 → 4-byte Write	PLB address decodes to PMM0, PMM1, or PMM2	Memory Write
Burst Write	PLB address decodes to PMM0, PMM1, or PMM2	Memory Write
Single-Beat 1 → 4-byte Read	Address 0x1 EED00000	Interrupt Acknowledge
Single-Beat 1 → 4-byte Write	Address 0x1 EED00000	Special Cycle
—	Not supported	Memory Write and Invalidate
—	Not supported	Memory Write Line

21.5.1.1 PCI Master Commands

The type of cycle generated to the PCI bus depends on the PLB address, the type of PLB transfer, and the data size. The following sections describe the transaction types supported and outlines the translation of commands from one bus to the other.

The terms "single beat" or "1–8-byte," in reference to PLB transfers, refer to the M_size=0000 transaction type.

PCI bridge initiates the following commands as a PCI master:

- I/O Read and I/O Write
  This command is generated in response to PLB 1–8-byte read or write requests that decode to one of the two PCI I/O spaces.

- Configuration Read and Configuration Write (type 0 and type 1)
  This command is generated in response to PLB 1–8-byte read or write requests that decode to the PCIC0_CFGDATA register.
- Memory Read
  This command is generated in response to PLB 1–8-byte reads or byte and half word burst reads that decode to one of the three PMMs when the PMM is marked as nonprefetchable.
- Memory Read Line
  This command is generated in response to PLB 4- and 8-word line reads or word and double word reads of 32 bytes or less that decode to one of the three PMMs.
- Memory Read Multiple
  This command is generated in response to PLB 1–8-byte reads or byte and half word burst reads that decode to one of the three PMMs when the PMM is marked as prefetchable. This command is also generated in response to word and double word burst reads of greater than 32 bytes that decode to one of the three PMMs. For prefetches, the PCI bridge bursts up to a 64 bytes from the PCI.
- Memory Write
  This command is generated in response to PLB 1–8-byte writes or burst writes to one of the three PMMs.
- Interrupt Acknowledge
  This command is generated in response to a PLB 1–8-byte read from address 0x1 EED00000.
- Special Cycle
  This command is generated in response to a PLB 1–8-byte write to address 0x1 EED00000.

The Memory Write and Invalidate command is not generated. All PCI memory writes are performed with Memory Write.

The PLB slave responds as a 64-bit device to word and double word bursts. All other commands receive a 32-bit response.

The PCI bridge supports PLB size 1–8-byte encodings. Burst reads of all sizes are also supported. Read line sizes greater than eight words are not supported, and no line writes are supported. The PCI bridge posts all writes which are decoded to PCI memory and PCI I/O space. Posted data is kept in internal write buffers until it can be transferred to the PCI bus. All other writes and all reads are connected, that is, they complete on the PCI bus before completing on the PLB.

21.5.1.2 PLB Slave Read Handling

PLB master read requests are decoded into four types: PCI Memory, I/O, Configuration, and Interrupt Acknowledge. If the request falls within any of these ranges, and is a supported command type, the bridge claims the cycle initially by asserting a PLB wait signal (as opposed to a PLB address acknowledge signal). The bridge must first gain access to the PCI bus before acknowledging a PLB read request. The specific timing of the address acknowledge is dependent upon the type of transfer. All posted writes must be flushed before a read is allowed to complete.

For PLB line reads, the PCI bridge must wait for all read data to be received before acknowledging the PLB request. This is because PCI targets are allowed to disconnect in the middle of a transfer, and the PLB requires line transfers to be atomic. If the system can guarantee that PCI targets do not disconnect these reads, PCIC0_BRDGOPT1[APLRM] can be set to 1. In this mode, line read performance is improved by having the bridge PLB slave assert an address acknowledge signal and begin its data tenure as soon as the first word is received on the PCI bus. If the above guarantee cannot be made, the setting of this bit could hang the bridge.

If the PCI cycle Master Aborts, all beats of read data are returned as 0xFFFFFFFF.

PLB master reads to the PCI bridge configuration registers are allowed to execute regardless of whether any write data is posted in the bridge. The configuration registers are described in *PCI Bridge Configuration Registers* on page 471.

21.5.1.3 Prefetching

When the PCI bridge receives a PLB 1–8-byte or word or double word burst read request that decodes to a PMM marked as nonprefetchable. The PCI bridge runs a single beat read to the PCI. If the PCI cycle is retried, the PLB cycle is rearbitrated.

When the PCI bridge receives a PLB 1–8-byte read request that decodes to a PMM marked as prefetchable, the PCI bridge burst reads up to a 64 bytes from the PCI and saves the data in the PLB slave read prefetch buffer. Less than 64 bytes can be read if the PCI target disconnects, or if the PCI bridge PCI master disconnects due to a master latency time out. Note that PCI bridge prefetching is not affected by memory management page boundaries (PLB_Guarded is ignored). If a subsequent PLB 1–8-byte or byte or half word burst read is contained in the prefetch buffer, the data is returned to the PLB directly from the prefetch buffer, and no cycle is generated on the PCI.

If a PLB read to the PCI bridge occurs while the PCI bridge is prefetching and does not hit in the prefetch buffer, then the PLB read is rearbitrated. After prefetching completes, any PLB read (of any type or address range) to the PCI bridge that does not hit in the prefetch buffer causes the prefetch buffer to be emptied, and a new PCI read to begin. PLB writes, including configuration writes, will invalidate the prefetch buffer.

21.5.1.4 PLB Slave Write Handling

PLB master write requests are decoded into four types: PCI Memory (one of three PMM ranges), PCI I/O, PCI Configuration, or Special Cycles. If the request falls within any of these ranges, and is a supported command type, the bridge claims the cycle by asserting a PLB wait signal. If the write is connected, or translates to a PCI Configuration or Special Cycle, the bridge must gain access to the PCI bus and successfully transfer the data before it may assert a PLB address acknowledge signal. If the address is to PCI I/O or memory, the bridge immediately asserts a PLB address acknowledge signal and posts the write if there is sufficient buffer space.

Internal configuration writes are not allowed to execute if posted write data exists in either the PCI slave write buffer or the PLB slave write buffer. The internal configuration mechanism is described in *PCI Bridge Configuration Registers* on page 471.

PLB Slave Write Post Buffer

The PCI bridge has a 32-byte write post buffer that may contain four separate single-beat PLB write transactions or one burst. New PLB write requests are rearbitrated if there is not enough room in the write post buffer.

The buffers are not snooped, and are always completed on the PCI bus in the same order as they are received on the PLB bus.

Each write buffer entry preserves the master ID and drives the appropriate PLB bus busy signal until the write is deallocated (it completes on the PCI bus). It is recommended that PLB masters do not use PLB bus busy signal. Instead, PLB masters generating cycles to the PCI should use the standard PCI mechanisms for data coherency.

21.5.1.5 Aborted PLB Requests

The PCI bridge aborts PLB reads only.

A PLB master accessing the PCI bridge can abort PLB write cycles only under the following conditions:

- The PCI bridge rearbitrates the cycle.
- The PCI bridge does not see the cycle because the PLB bus is granted to some other master. A CPU/System Memory interface is expected to do this when a CPU cycle is pending to PCI bridge, but a PLB Master requests system memory access requiring snooping.

Note: If a PLB master aborts the write cycle at any other time, the results are undefined and the bus may hang.

21.5.1.6 Retried PCI Reads

The PCI specification requires that a PCI master must repeat any read that is retried. The PCI bridge adheres to this requirement. It is only mentioned here because, under certain conditions with respect to aborted PLB reads, the PCI bridge must execute a PCI read and discard the data.

21.5.2 PCI-to-PLB Transaction Handling

This section describes how PCI bridge handles read and write requests from a PCI master device. PCI bridge responds as a PCI target to PCI memory transactions when the PCI address is in one of the two PTM ranges and PCIC0_CMD[MA] = 1. PCI bridge responds by claiming the PCI cycle and mastering a cycle on the PLB.

PCI bridge is also a PCI target for configuration cycles when its PCIIDSel pin is active. PCI bridge will master abort if a configuration cycle is run to itself.

Table 21-6. Transaction Mapping: PCI → PLB

PCI Transaction PCI Master → Bridge (PCI Target Interface)	Bridge Mapping and Qualifications	PLB Transaction Bridge (PLB Master Interface) → PLB Slave
Single-Beat Memory Read	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	8-byte or double word burst read
Burst Memory Read	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	8-byte or double word burst read
Memory Read Line	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst read
Memory Read Multiple	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst read
Memory Read	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst read
Single-Beat Memory Write	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	1 → 8-byte write
Single-Beat Memory Write and Invalidate	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	1 → 8-byte write
Burst Memory Write	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst write
Burst Memory Write and Invalidate	PCI address decodes to PTM1/BAR1 or PTM2/ BAR2, memory access flag	Double word burst write
—	Not supported	Memory line reads

Table 21-6. Transaction Mapping: PCI → PLB (continued)

PCI Transaction PCI Master → Bridge (PCI Target Interface)	Bridge Mapping and Qualifications	PLB Transaction Bridge (PLB Master Interface) → PLB Slave
—	Not supported	Memory line writes

21.5.2.1 PLB Master Commands

PCI bridge generates PLB transactions based on the type and length of received PCI transactions. The following sections describe the transaction types supported and outline the translation of commands from one bus to the other.

The term "single-beat" refers to the M_size = 0000 transaction type. PCI slave devices are referred to as "targets."

PCI bridge initiates the following PLB master commands:

- 8-Byte Read:
  Generated in response to single beat or burst Memory Read commands from the PCI bus.
- Double word Burst Read:
  Generated in response to Memory Read Line and Memory Read Multiple commands on the PCI bus. PCI bridge can also be programmed to perform double word bursts on behalf of Memory Read.
- 1–8-Byte Write:
  Generated in response to single-beat (1–4 byte) Memory Write commands on the PCI bus; also generated when the PCI master uses non-contiguous byte enables (see *Byte Enable Handling* on page 469).
- Double word Burst Write:
  Generated in response to burst Memory Write and Memory Write and Invalidate commands on the PCI bus.

The PLB treats Memory Write and Memory Write and Invalidate identically (nothing on the PLB distinguishes a Memory Write from a Memory Write and Invalidate.)

PCI bridge does not generate line reads or line writes on the PLB.

21.5.2.2 Handling of Reads from PCI Masters

PCI bridge responds to PCI Memory Read, Memory Read Line, and Memory Read Multiple commands. The PCI bridge initiates all PLB reads as single-beat or double word burst transfers.

Memory Read generates a PLB single-beat double word read. Memory Read Line and Memory Read Multiple commands generate PLB double word bursts. For Memory Read Line, PCI bridge encodes a burst length on the byte enable pins of the PLB that corresponds to the number of double words from the start address to the end of a word boundary and terminates when the encoded number of words has been transferred. This is called a PLB fixed length burst. If the starting address is the last double word on a word boundary (typically, this should not occur), PCI bridge executes a single-beat read. For Memory Read Multiple, PCI bridge sets the byte enables to 0s, indicating a variable length burst.

The PCI target can be programmed to treat Memory Reads as Memory Read Line commands or Memory Read Multiple commands in terms of PLB read behavior.

The PCI bridge guarantees the PCI initial target latency requirement by retrying the PCI cycle if read data is not immediately available in the read buffer. Subsequent latency is programmable using PCIC0_BRDGOPT2[STLD].

The PCI bridge master latency timer can limit the length of read bursts using PCIC0_BRDGOPT1[MLTC]. The timer limits the duration of a burst to the programmed value in units of PLB clock cycles.

Read Buffer

The PCI bridge read buffer stores all read data (including delayed read and prefetched data) when the data is received from the PLB, before it is passed to the PCI. The 96-byte read buffer can store one transaction.

Delayed Reads

A delayed read is queued if a PCI master requests a read while PCI master writes are posted. Posted writes are completed on the PLB before the read is run. PCI bridge continues to post PCI master writes (if buffer space is available) while a delayed read is in progress. Such writes complete on the PLB after the read, even though they complete on the PCI before the read.

A delayed read is also queued if data is not immediately available in the read buffer and a delayed read does not already exist.

When a PCI master returns for a previously requested delayed read, the data is passed out of the read buffer. While the PCI master accepts delayed read data, the PCI bridge can begin to prefetch more read data, if the PCI master posted write buffer is empty. See *Read Prefetching* on page 468 for more details.

Any data remaining in the read buffer after delayed read data has been passed to a PCI master is marked as prefetch data and discarded upon a write in either direction.

PCI bridge can hold one delayed read transaction. PCI bridge retries all other PCI master reads until the delayed read completes on the PCI. The read buffer discards data from a delayed read under only one condition. The PCI discard timer is used to track the amount of time it takes for a PCI master to re-request the read. If the PCI master does not re-request the read in 2^{15} PCI clocks (about one millisecond for a 33 MHz PCI clock), PCI bridge discards the delayed read data. This timer begins counting at the beginning of the initial PCI cycle (delayed read request). If PCI bridge is used in a system on which the PLB target (memory) maximum latency (including PLB arbitration) is a significant portion of the timer duration, the timer can expire despite normal bus operation. One solution to this problem is to disable the PCI Discard Timer.

If a delayed read is burst terminated on the PLB (a rare occurrence), PCI bridge will not repeat the request until the PCI master re-requests and only then if the PCI master requests more data than is already buffered.

Read Prefetching

PCI bridge attempts to prefetch data to maximize burst throughput on PCI read requests. Read prefetching occurs only in response to Memory Read Multiple commands or Memory Read, if the PCI target is programmed to treat them as Memory Read Multiple (Memory Read Line causes prefetching to the next word boundary only). This prefetch buffer is 96 bytes.

If a PCI master reads from the read buffer while a PLB read is in progress, data is passed to PCI as it is being filled from the PLB. If the read buffer goes empty long enough for the PCI subsequent latency timer to expire, the PCI is target disconnected. If the read buffer fills up, the PLB cycle is master terminated. The bridge PLB master will not attempt to reacquire the PLB bus if its posted write buffer is not empty.

Prefetched data is discarded if a write is accepted from either the PLB or the PCI. A PCI master read that misses the prefetch buffer also causes current read data to be discarded and the new request to be serviced.

Byte Enable Handling

PCI byte enables are treated as don't cares for PCI reads. The PCI bridge performs double word burst or single-beat doubleword reads on the PLB, regardless of the byte enables presented by the requesting PCI master.

Note: This rule assumes that all PLB memory is prefetchable and that all PLB memory accesses are nondestructive.

Handling Writes from PCI Masters

PCI bridge responds to Memory Write and Memory Write and Invalidate commands. All PCI master writes are posted. A 64-byte write buffer is used for this purpose. The write buffer accepts up to two separate PCI write transactions. Two single-beat writes, one burst write, or a combination of a single-beat and a burst writes can be posted. If the write buffer is full, new writes are retried until buffer space becomes available.

Note: The maximum of two posted transactions is as viewed from the PCI master. The number of writes performed on the PLB can be more than two, depending on the setting of byte enables of write burst data. See *Byte Enable Handling* on page 469.

The PCI bridge begins a PLB write request as soon as a PCI master write has completed on the PCI bus, or a bursting PCI master has written at least six words of data. The PCI bridge continues to receive data from a bursting PCI master as it transfers data to the PLB. If the write post buffer fills, the PCI master is target disconnected. If the write post buffer empties, the PLB cycle is master terminated.

Writes are executed in the same order they are received.

Byte Enable Handling

The PLB does not support non-contiguous byte enables, whereas the PCI bus does. The PLB supports the use of byte enables only for non-line, non-burst transactions, whereas the PCI bus supports any combination of byte enables for any data phase. Therefore, when a PCI master presents a data phase without all byte enables asserted, the bridge disconnects and treats that data phase as one or two single-beat writes on PLB, depending on whether or not byte enables are non-contiguous.

Masters presenting writes without all byte enables asserted experience degraded performance.

21.5.2.3 Miscellaneous

The PCI target forces single-beat transfers when reserved burst or cache line wrap order is used. The PCI target causes master abort of reserved command encodings, and does not respond to I/O, interrupt acknowledge, or special cycle commands. The PLB master does not abort requests.

21.5.3 Completion Ordering

PCI bridge implements the following completion ordering rules:

1. PCI master writes are accepted if there is room in the PCI write post buffer.
2. New PCI master reads are accepted if there is no delayed read (DRR or DRC) in progress:
   a. If PCI write post buffer is empty and read data is not buffered, then begin a delayed read (enter DRR state).
   b. If PCI write post buffer is not empty, then begin delayed read (enter DRR state).

   Delayed reads are handled as follows:
   a. While in DRR state, retry all PCI master reads. Wait for all PCI master writes if any that were posted before entering DRR state to complete on PLB.
   b. Execute PLB read, enter DRC state.
   c. While in DRC state, retry all PCI master reads if the address does not match. If it does match, pass the read data to the PCI master. If data is passed, exit the DRC state.
3. PLB master writes are accepted if there is room in the PLB write post buffer.
4. PLB master reads are accepted if the PLB write post buffer and the PCI write post buffer are both empty.

21.5.3.1 PCI Producer-Consumer Model

The PCI Producer-Consumer model is followed with one exception: PCI master reads do not flush PLB writes and PCI master writes do not cause PLB prefetched read data to be discarded. If the "flag" is stored in system memory (PLB side), but the "data" is stored in a PCI target, the control software must manually force coherency. This can be done by following two rules:

1. To ensure data written by a PLB master has reached the intended PCI target, the PLB master should execute a read from PCI, to any nondestructive address. This is only necessary if the write is postable.
2. To ensure data read by a PLB master is current (rather than old prefetched data), the PLB master should execute a read from PCI to any other nondestructive address. This is only necessary if the read is prefetchable.

21.5.4 Collision Resolution

The PCI bridge must resolve collisions when a PLB master and a PCI master attempt accesses through the PCI bridge at the same time. *Table 21-7* summarizes collision resolution.

In general, PLB postable writes are always accepted (if buffer space is available), and passed to the PCI when given the chance. PCI master writes are always accepted (if buffer space is available), but cause PLB reads and non-postable writes to be rearbitrated to clear the path to the PLB. PLB reads and non-postable writes proceed as long as there is no PCI master activity, which causes the PLB cycle to be rearbitrated. PCI master reads are always allowed to proceed, and cause PLB reads and non-postable writes to be rearbitrated.

Internal configuration accesses have their own rules. Configuration writes are not allowed to complete while any write data is posted in the PCI bridge, or while the PCI master is prefetching. Otherwise, there are no restrictions. Configuration reads have no restrictions; however, only one internal configuration access (PLB or PCI side) may be serviced at a time.

Table 21-7. Collision Resolution

Access Type	PLB Read from PCI	PLB Postable Write to PCI	PLB Non-postable Write to PCI
PCI Write to PLB	Rearbitrate PLB master (reads flush writes)	No conflict	Rearbitrate PLB master
PCI Read from PLB	Rearbitrate PLB master	No conflict	Rearbitrate PLB master

21.5.5 PCI Control Register (SDR0_PCI0)

The following figure describes the SDR0_PCI0 register bits.

Figure 21-7. PCI Control Register (SDR0_PCI0)

0	PAE	PCI Arbiter Enable 0 PCI arbiter disabled 1 PCI arbiter enabled	Reset value = SDR0_SDSTP1[PAE]
1	PHCE	PCI Host Configuration Enable 0 PCI host configuration disabled 1 PCI host configuration enabled	Reset value = SDR0_SDSTP1[PHCE]
2	PAME	PCI Asynchronous Mode Enable 0 Reserved 1 PCI asynchronous mode enabled	Reset value = SDR0_SDSTP1[PAME] PAME = 0 is not supported.
3:31		Reserved	Reset value = 0

21.6 PCI Bridge Configuration Registers

The PCI bridge has two sets of configuration registers for configuring the bridge, handling errors, and reporting status. Local configuration registers control PLB functions, and are accessed only from the PLB. The PCI configuration registers control PCI functions, and can be accessed from the PLB and the PCI. In addition, the mechanism used to access the bridge configuration registers may also be used to configure other devices on the PCI bus.

21.6.1 PCI Bridge Register Summary

Table 21-8 provides a summary of all of the PCI bridge registers. The registers are discussed in detail in the following sections. See *Reset and Initialization* on page 169 for register reset values.

Table 21-8. Directly Accessed MMIO Registers

Register	Address	Access	Description	Page
PCIL0_PMM0LA	0x1 EF40 0000	R/W	PMM 0 Local Address	473
PCIL0_PMM0MA	0x1 EF40 0004	R/W	PMM 0 Mask/Attribute	475
PCIL0_PMM0PCILA	0x1 EF40 0008	R/W	PMM 0 PCI Low Address	474
PCIL0_PMM0PCIHA	0x1 EF40 000C	R/W	PMM 0 PCI High Address	474
PCIL0_PMM1LA	0x1 EF40 0010	R/W	PMM 1 Local Address	474
PCIL0_PMM1MA	0x1 EF40 0014	R/W	PMM 1 Mask/Attribute	473
PCIL0_PMM1PCILA	0x1 EF40 0018	R/W	PMM 1 PCI Low Address	475
PCIL0_PMM1PCIHA	0x1 EF40 001C	R/W	PMM 1 PCI High Address	475
PCIL0_PMM2LA	0x1 EF40 0020	R/W	PMM 2 Local Address	475
PCIL0_PMM2MA	0x1 EF40 0024	R/W	PMM 2 Mask/Attribute	475
PCIL0_PMM2PCILA	0x1 EF40 0028	R/W	PMM 2 PCI Low Address	476
PCIL0_PMM2PCIHA	0x1 EF40 002C	R/W	PMM 2 PCI High Address	476
PCIL0_PTM1MS	0x1 EF40 0030	R/W	PTM 1 Memory Size/Attribute	476
PCIL0_PTM1LA	0x1 EF40 0034	R/W	PTM 1 Local Address	476
PCIL0_PTM2MS	0x1 EF40 0038	R/W	PTM 2 Memory Size/Attribute	477
PCIL0_PTM2LA	0x1 EF40 003C	R/W	PTM 2 Local Address	477

Table 21-9. PCI Configuration Address and Data Registers

Register	Address	Access	Description
PCIC0_CFGADDR	0x1 EEC0 0000	R/W	PCI Configuration Address Register
PCIC0_CFGDATA	0x1 EEC0 0004	R/W	PCI Configuration Data Register

Table 21-10. PCI Configuration Register Addresses

Register	Address (See Note)	Access		Description
		PLB	PCI	
PCIC0_VENDID	0x8000 0000	R/W	R	PCI Vendor ID
PCIC0_DEVID	0x8000 0002	R/W	R	PCI Device ID
PCIC0_CMD	0x8000 0004	R/W	R/W	PCI Command Register
PCIC0_STATUS	0x8000 0006	R/W	R/W	PCI Status Register
PCIC0_REVID	0x8000 0008	R/W	R/W	PCI Revision ID
PCIC0_CLS	0x8000 0009	R/W	R	PCI Class Register
PCIC0_CACHELS	0x8000 000C	R	R	PCI Cache Line Size
PCIC0_LATTIM	0x8000 000D	R/W	R/W	PCI Latency Timer
PCIC0_HDTYPE	0x8000 000E	R	R	PCI Header Type
PCIC0_BIST	0x8000 000F	R	R	PCI Built In Self Test Control
PCIC0_PTM1BAR	0x8000 0014	R/W	R/W	PCI PTM 1 BAR
PCIC0_PTM2BAR	0x8000 0018	R/W	R/W	PCI PTM 2 BAR
PCIC0_SBSYSVID	0x8000 002C	R/W	R	PCI Subsystem Vendor ID
PCIC0_SBSYSID	0x8000 002E	R/W	R	PCI Subsystem ID
PCIC0_CAP	0x8000 0034	R	R	PCI Capabilities Pointer
PCIC0_INTLN	0x8000 003C	R/W	R/W	PCI Interrupt Line
PCIC0_INTPN	0x8000 003D	R	R	PCI Interrupt Pin
PCIC0_MINGNT	0x8000 003E	R	R	PCI Minimum Grant
PCIC0_MAXLTNCY	0x8000 003F	R	R	PCI Maximum Latency
PCIC0_ICS	0x8000 0044	R/W	R/W	PCI Interrupt Control/Status
PCIC0_ERREN	0x8000 0048	R/W	R/W	Error Enable
PCIC0_ERRSTS	0x8000 0049	R/W	R/W	Error Status
PCIC0_BRDGOPT1	0x8000 004A	R/W	R/W	PCI Bridge Options 1
PCIC0_PLBBESR0	0x8000 004C	R/W	R/W	PLB Slave Error Syndrome 0
PCIC0_PLBBESR1	0x8000 0050	R/W	R/W	PLB Slave Error Syndrome 1
PCIC0_PLBBEAR	0x8000 0054	R/W	R/W	PLB Slave Error Address Register
PCIC0_CAPID	0x8000 0058	R	R	Capability Identifier
PCIC0_NEXTIPTR	0x8000 0059	R	R	Next Item Pointer
PCIC0_PMC	0x8000 005A	R	R	Power Management Capabilities
PCIC0_PMCSR	0x8000 005C	R/W	R/W	Power Management Control Status
PCIC0_PMCSRBSE	0x8000 005E	R	R	PMCSR PCI-to-PCI Bridge Support Extensions
PCIC0_DATA	0x8000 005F	R	R	Data

Table 21-10. PCI Configuration Register Addresses (continued)

Register	Address (See Note)	Access		Description
		PLB	PCI	
PCIC0_BRDGOPT2	0x8000 0060	R/W	R/W	PCI Bridge Options 2
PCIC0_PMSCRR	0x8000 0064	R/W	R/W	Power Management State Change Request Register
Note: Use this value in PCIC0_CFGADDR.				

21.6.2 PCI Bridge Local Configuration Registers

The PCI bridge local configuration registers have fixed addresses in PLB space and must be accessed using single beat PLB read or write cycles of the same size as shown in the register descriptions.

Failure to access all bytes of a particular register could produce unexpected results. Reading of reserved bit locations produces unpredictable values. Software must use appropriate masks to extract the desired bits. Writes must preserve the values of reserved bit positions by first reading the register, merging the new value, and writing the result.

21.6.2.1 PMM 0 Local Address Register (PCIL0_PMM0LA)

PCIL0_PMM0LA defines the PLB starting address of range 0 in PLB space that is mapped to PCI memory. Only bits that are 1 in the PCIL0_PMM0MA are used to determine the starting address; all other bits are don't cares. Only bits 31:12 are writable. Bits 11:0 are always 0.

Figure 21-8. PMM 0 Local Address Register (PCIL0_PMM0LA)

31:12	WLA	Writable PLB Local Address	
11:0		PLB Local Address	Always 0

21.6.2.2 PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)

PCIL0_PMM0MA controls the size and attributes of the PLB space mapped to PCI memory for range 0.

Figure 21-9. PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)

31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM0LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as all ones.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable Memory mapping is enabled.	Note that PCIL0_PMM0LA, PCIL0_PMM0PCIHA, and PCIL0_PMM0PCILA must be initialized before enabling.

21.6.2.3 PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)

PCIL0_PMM0PCILA defines the low-order 32 bits of the PCI address generated in response to a PLB access to range 0. Only bits that are 1 in PCIL0_PMM0MA are passed to the PCI address. The other (least significant) bits of the PCI address are passed through from the PLB address. Only bits 31:12 are writable. Bits 11:0 are always 0.

Figure 21-10. PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)

31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

21.6.2.4 PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)

PCIL0_PMM0PCIHA defines the high-order 32 bits of the PCI address generated in response to a PLB access to range 0. If PCIL0_PMM0PCIHA is greater than 0, the PCI bridge generates a PCI dual address cycle using the value in PCIL0_PMM0PCIHA as the high-order 32 bits of the PCI address.

Figure 21-11. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA)

31:0		PCI High Address	

21.6.2.5 PMM 1 Local Address Register (PCIL0_PMM1LA)

PCIL0_PMM1LA defines the PLB starting address of range 1 in PLB space that is mapped to PCI memory. See *PMM 0 Local Address Register (PCIL0_PMM0LA)* on page 473.

Figure 21-12. PMM 1 Local Address Register (PCIL0_PMM1LA)

31:0		PLB Local Address	

21.6.2.6 PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)

PCIL0_PMM1MA defines the size and attributes of range 1 in PLB space that is mapped to PCI memory. See *PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)* on page 473.

Figure 21-13. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)

31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM1LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as 0x11111.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.

0	ENA	PLB to PCI Memory Mapping Enable Memory mapping is enabled.	Note that PCIL0_PMM1LA, PCIL0_PMM1PCIHA, and PCIL0_PMM1PCILA must be initialized before enabling.

21.6.2.7 PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)

PCIL0_PMM1PCILA defines the low-order 32 bits of the PCI address generated in response to a PLB access to range 1. See *PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)* on page 474.

Figure 21-14. PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)			
31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

21.6.2.8 PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)

PCIL0_PMM1PCIHA defines the high-order 32 bits of the PCI address generated in response to a PLB access to range 1. See *PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)* on page 474.

Figure 21-15. PMM 0 PCI High Address Register (PCIL0_PMM1PCIHA)			
31:0		PCI High Address	

21.6.2.9 PMM 2 Local Address Register (PCIL0_PMM2LA)

PCIL0_PMM2LA defines the PLB starting address of range 2 in PLB space that is mapped to PCI memory. See *PMM 0 Local Address Register (PCIL0_PMM0LA)* on page 473.

Figure 21-16. PMM 2 Local Address Register (PCIL0_PMM2LA)			
31:0		PLB Local Address	

21.6.2.10 PMM 2 Mask/Attribute Register (PCIL0_PMM2MA)

PCIL0_PMM2MA defines the size and attributes of range 2 in PLB space that is mapped to PCI memory. See *PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)* on page 473.

Figure 21-17. PMM 2 Mask/Attribute Register (PCIL0_PMM2MA)			
31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM2LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as 0x11111.
11:2		Reserved	Returns 0 when read.

1	PRE	Read Prefetching Enable Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable Memory mapping is enabled.	Note that PCIL0_PMM2LA, PCIL0_PMM2PCIHA, and PCIL0_PMM2PCILA must be initialized before enabling.

21.6.2.11 PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)

PCIL0_PMM2PCILA defines the low-order 32 bits of the PCI address generated in response to a PLB access to range 2. See *PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)* on page 474.

Figure 21-18. PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)

31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

21.6.2.12 PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)

PCIL0_PMM2PCIHA defines the high-order 32 bits of the PCI address generated in response to PLB access to range 2. See *PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)* on page 474.

Figure 21-19. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)

31:0		PCI High Address	

21.6.2.13 PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)

PCIL0_PTM1MS defines the size and attributes of the of PCI memory region mapped to local (PLB) space through PTM 1. PCIL0_PTM1MS affects the operation of PCI PTM 1 BAR.

Figure 21-20. PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)

31:12	SIZE	Defines the size of the region of PCI memory space that is mapped to local (PLB) space using PTM 1.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
11:1		Reserved	Returns 0 when read.
0	ENA	Determines if range 1 is enabled to map PCI memory space to PLB space.	

21.6.2.14 PTM 1 Local Address Register (PCIL0_PTM1LA)

PCIL0_PTM1LA defines the local (PLB) address that is generated in response to a PCI access to local (PLB) space through PTM 1. Only bits that are 1 in PCIL0_PTM1MS are passed to the PLB address. The other (least significant) bits of the PLB address are passed through from the PCI address. Only bits 31:12 are writable. Bits 11:0 are always 0.

Figure 21-21. PTM 2 Local Address Register (PCIL0_PTM1LA)			
31:12	WLA	Writable PTM 1 Local Address	Writable
11:0		PTM 1 Local Address	Always 0

21.6.2.15 PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)

PCIL0_PTM2MS defines the size of the region of PCI memory space mapped to local (PLB) space through PTM 2.

Figure 21-22. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)			
31:12	SIZE	Defines the size of the region of PCI memory space mapped to local (PLB) space using PTM 2.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
11:1		Reserved.	Returns 0 when read.
0	ENA	Determines if range 2 is enabled to map PCI memory space to PLB space.	When ENA is disabled, PCIC0_PTM2BAR cannot be written. Set PCIC0_PTM2BAR to 0 before disabling ENA.

21.6.2.16 PTM 2 Local Address Register (PCIL0_PTM2LA)

This register defines the local (PLB) address generated in response to a PCI access to local (PLB) space through PTM 2. See *PTM 1 Local Address Register (PCIL0_PTM1LA)* on page 476 for more information.

Figure 21-23. PTM 2 Local Address Register (PCIL0_PTM2LA)			
31:0		PTM 2 Local Address	

21.6.3 PCI Configuration Registers

The PCI configuration registers can be accessed from both the PLB and PCI buses. PLB side configuration is supported using the mechanism defined in the *PCI Local Bus Specification* Version 2.2. This mechanism uses PCIC0_CFGADDR and PCIC0_CFGDATA to access the configuration registers indirectly. These registers reside at addresses 0x1 EEC00000 and 0x1 EEC00004, respectively. To access (from the PLB side) the configuration space of other devices on the PCI bus, write a value to PCIC0_CFGADDR that specifies the following:

- Bus number
- Device number on that bus
- Register number to be accessed

The value must also set PCIC0_CFGADDR[EN] = 1. An access to PCIC0_CFGDATA then results in a configuration cycle on the PCI bus.

To access the bridge configuration registers from the PLB side, use the same mechanism as described above, but set PCIC0_CFGADDR[BN, DN] = 0. The bridge is assumed to reside on PCI bus 0 and to have a device number of 0.

The bridge configuration registers can be accessed from the PCI side by Type 0 configuration reads or writes, with the PCIIDSel pin asserted to the bridge. There are some restrictions on PCI side accesses that are noted in the register descriptions that follow.

PCIC0_CFGADDR and CONFIG_DATA should be accessed with single-beat PLB commands. All registers are byte addressable. Reading of reserved bit locations produces unpredictable values. Software must use appropriate masks to extract the desired bits. Writes must preserve the values of reserved bit positions by first reading the register, merging the new value, and writing the result.

21.6.3.1 PCI Configuration Address Register (PCIC0_CFGADDR)

PCIC0_CFGADDR controls the type of cycle generated when PCIC0_CFGDATA is accessed. Its fields are shown in *Figure 21-24*.

Figure 21-24. PCI Configuration Address Register (PCIC0_CFGADDR)

31	EN	Enable 0 Disabled 1 Enabled	
30:24		Reserved	
23:16	BN	Bus Number	
15:11	DN	Device Number	
10:8	FN	Function Number	
7:2	RN	Register Number	
1	0		
0	0		

See the *PCI Local Bus Specification* Version 2.2 for details about how the fields are used.

21.6.3.2 PCI Configuration Data Register (PCIC0_CFGDATA)

Accessing PCIC0_CFGDATA causes one of three things to happen, depending on the value of PCIC0_CFGADDR.

1. Generation of a Type 0 configuration cycle on the PCI bus (PCIC0_CFGADDR[BN] = 0, PCIC0_CFGADDR[DN] > 0).
2. Generation of a Type 1 configuration cycle on the PCI bus (PCIC0_CFGADDR[BN] > 0).
3. Access of a PCI bridge PCI configuration register (PCIC0_CFGADDR[BN, DN] = 0).

Figure 21-25 illustrates the PCIC0_CFGDATA register.

Figure 21-25. PCI Configuration Data Register (PCIC0_CFGDATA)

31:0		Configuration Data	

21.6.3.3 PCI Vendor ID Register (PCIC0_VENDID)

PCIC0_VENDID identifies the manufacturer of a PCI device. This register contains 0x1014 (index 0x00 = 0x14, index 0x01 = 0x10) at reset. This vendor ID is assigned for all PCI devices. The local CPU (PLB master) can write a different value to this register.

Figure 21-26. PCI Vendor ID Register (PCIC0_VENDID)

15:0		Vendor ID	

21.6.3.4 PCI Device ID Register (PCIC0_DEVID)

PCIC0_DEVID identifies the PCI device. This value is 0x027F (index 0x03 = 0x02, index 0x02 = 0x7F) at reset at reset. The local CPU (PLB master) can write a different value to this register.

Figure 21-27. PCI Device ID Register (PCIC0_DEVID)

15:0		PCI Device ID	

21.6.3.5 PCI Command Register (PCIC0_CMD)

PCIC0_CMD controls the operation of the PCI bridge on the PCI bus. *Figure 21-28* describes the bits.

Figure 21-28. PCI Command Register (PCIC0_CMD)

15:10		Reserved	.
9	FBB	Fast Back-to-Back Write Enable	Enables PCI masters to perform fast back-to-back transactions. Because he PCI bridge does not perform fast back-to-back transactions; FBB is read-only and returns 0 when read.
8	SE	$\overline{\text{PCISErr}}$ Enable 0 Disabled 1 Enabled	Enables driving $\overline{\text{PCISErr}}$ when a PCI bus parity error is detected when the PCI bridge is the PCI target. PCIC0_CMD[PER] must be enabled for address parity errors. PCIC0_CMD[PER] and PCIC0_ERREN[WDPE] must be enabled for write data parity errors.
7	AS	Address stepping wait states.	The PCI bridge does not address step (except for address stepping when generating a Config Type 0 cycle); AS is read-only and returns 0 when read.
6	PER	Parity error response 0 Disabled 1 Enabled	This bit is enabled for all types of PCI bus parity errors, including the following: • PCI data bus parity errors while PCI is master. • PCI data bus parity errors while PCI is target. • PCI address bus parity errors. When parity error response is disabled, detection of these errors is masked and $\overline{\text{PCIPErr}}$ (PERR#) is not asserted, although parity is still generated.
5	PS	Palette Snooping	Enable special palette snooping. The PCI bridge is not a VGA device; PS is read-only and returns 0 when read
4	MWI	Memory Write and Invalidate Enable	The PCI bridge does not generate this command; MWI is read-only and returns 0 when read.

3	SC	Special Cycle Operations Enable	The PCI bridge never monitors special cycles; SC is read-only and returns 0 when read.
2	ME	Master Enable 0 Disabled 1 Enabled	Enables PCI bridge-to-master cycles on the PCI bus. When ME is 0, the PCI bridge only responds as a PLB slave to PCIC0_CFGADDR, PCIC0_CFGDATA, and PCI bridge local configuration register access. Except for configuration cycles, the PCI bridge cannot master cycles to the PCI bus. If the pin strapping setting reflected in CPC0_PSR[RL] = 1, ME resets to 1.
1	MA	Memory Access 0 Disabled 1 Enabled	Controls PCI bridge response as a PCI memory target. MA is disabled at reset.
0	IOA	I/O Access	Controls the PCI bridge response as a PCI I/O target. The PCI bridge does not respond to I/O space accesses; IOA is read-only and returns 0 when read.

21.6.3.6 PCI Status Register (PCIC0_STATUS)

PCIC0_STATUS is a read/bit-reset register used to record status information for PCI bus events. Bits in PCIC0_STATUS are set only as a result of specific events occurring on the PCI bus. They are reset by writing a 1 to the desired bit. Writing a 0 to a bit location leaves that bit unchanged.

Figure 21-29. PCI Status Register (PCIC0_STATUS)

15	DEPE	Detected Parity Error Write 1 to clear.	The PCI bridge sets DEPE when the PCI bridge detects a PCI bus parity error, regardless of the setting of any enable bits (DEPE is non-maskable). The following events set DEPE: PCI address bus parity error detected when PCI bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PCI bridge is the target). PCI data bus parity error detected when PCI bridge masters a PCI read cycle.
14	SSE	Signaled System Error Write 1 to clear.	The PCI bridge sets SSE if the PCI bridge asserts PCISErr (see *Error Handling* on page 493 for causes of PCISErr assertion).
13	RMA	Received Master Abort Write 1 to clear.	The PCI bridge sets RMA when a PCI cycle for which the PCI bridge is the master is terminated with master abort.
12	RTA	Received Target Abort Write 1 to clear.	The PCI bridge sets RTA when a PCI cycle for which it is the master is terminated with target abort.
11	STA	Signaled Target Abort Write 1 to clear.	The PCI bridge sets STA when a PCI cycle for which it is the target is terminated with target abort.
10:9	DST	$\overline{\text{PCIDevSel}}$ Response Timing Read-only.	The PCI bridge asserts $\overline{\text{PCIDevSel}}$ on the second clock after PCIFframe is asserted (called medium response time). Read-only; always returns 0b01 when read.

8	DPE	Data Parity Error Detected Write 1 to clear.	DPE is set when the following conditions are met: The PCI bridge detects a data parity error (PCIPErr is asserted) when the PCI bridge is the master on a PCI read cycle, or is the master when it samples PCIPErr asserted on a PCI write cycle. PCIC0_CMD[PER] = 1.
7	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Indicates that the PCI target can accept fast back-to-back transactions when the transactions are not to the same agent. The PCI bridge target does not accept this type of fast back-to-back transaction.
6	UDFS	UDF Supported Read-only; returns 0 when read.	Indicates device support of user-definable features. The PCI bridge does not support user-definable features.
5	66C	66 MHz Capable At reset PCI bridge is configured for 66MHz operation.	Indicates that the device can run at 66 MHz. The PCI bridge can be configured to run at 33 MHz max or 66 MHz. The local CPU (PLB master) sets 66C to 1 if PCI bridge is configured for 66 MHz operation.
4	CL	Capabilities List This bit is read only and returns 1 when read.	Indicates that the value at offset 0x34 is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits return 0s when read.

21.6.3.7 PCI Revision ID Register (PCIC0_REVID)

PCIC0_REVID holds the current incremental revision number. The reset value is the version number of the PCI bridge macro (current version is 20). However, the local CPU (PLB master) can write a value to this register.

Figure 21-30. PCI Revision ID Register (PCIC0_REVID)

7:0		Revision ID	Revision level of device.

21.6.3.8 PCI Class Register (PCIC0_CLS)

This register holds the class code. This register is 0x068000 at reset, which indicates that the PCI bridge is a bridge device located between the PLB and the PCI bus; however, the local CPU (PLB master) can write a value to this register for the case where PCI bridge is not the host bridge.

Class information is defined in the *PCI Local Bus Specification*, Version 2.2.

Figure 21-31. PCI Class Register (PCIC0_CLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device.
15:8	SUB	Subclass	Reset to 0x80, which indicates other bridge type.
7:0	INT	Interface Class	Reset to 00.

21.6.3.9 PCI Cache Line Size Register (PCIC0_CACHELS)

PCIC0_CACHELS determines the size of a PCI cache line. PCI bridge does not support a PCI cache. Therefore, this register is read-only and returns 0x00 when read.

Figure 21-32. PCI Cache Line Size Register (PCIC0_CACHELS)

7:0		PCI Cache Line Size	

21.6.3.10 PCI Latency Timer Register (PCIC0_LATTIM)

PCIC0_LATTIM holds the value of the PCI latency timer. The granularity of the latency timer is 8 PCI cycles. POR value is 0x00.

Figure 21-33. PCI Latency Timer Register (PCIC0_LATTIM)

7:0		PCI Latency Timer	

PCI Local Bus Specification Version 2.2 specifies that PCI masters capable of multi-beat bursts must, after losing their PCI grant, get off the bus when PCIC0_LATTIM has decremented to 0.

The actual number of clock cycles to disconnect varies somewhat when in asynchronous mode. If PCIC0_LATTIM is programmed in asynchronous mode to a value that is less than 64, the PCI bridge PCI master interface could timeout, regardless of the state of its grant line.

In asynchronous mode, the PCI master starts its timer and can timeout regardless of the state of the PCI grant. This is strictly a performance issue and does not limit functionality or affect data integrity.

Several factors affect the frequency of time outs.

- The amount of PCI bus traffic. In moderate to heavily loaded systems, this is less of an issue because a PCI master tends to lose its grant more often after gaining the bus.
- Fast targets that introduce few or no wait states reduce the chances of time outs occurring.

21.6.3.11 PCI Header Type Register (PCIC0_HDTYPE)

PCIC0_HDTYPE (bits 0:6) identifies the second part of the PCI header, which begins at offset 0x10. It also determines whether a device contains multiple functions (bit 7). The PCI bridge implements the standard header and is not a multi-function device; therefore, PCIC0_HDTYPE is read-only and returns 0x00 when read.

Figure 21-34. PCI Header Type Register (PCIC0_HDTYPE)

7:0		PCI Header Type	

21.6.3.12 PCI Built-In Self Test (BIST) Control Register (PCIC0_BIST)

PCIC0_BIST is used for control and status of BIST. PCI bridge does not implement BIST. PCIBIST is read-only and returns 0x00 when read.

Figure 21-35. PCI Built-in Self Test Control Register (PCIC0_BIST)			
7:0		PCI BIST Control	

21.6.3.13 Unused PCI Base Address Register Space

PCI base address register space is defined to begin at offset 0x10; however, the first 32 bits of this space are unused by PCI bridge, and the defined base address registers begin at offset 0x14.

21.6.3.14 PCI PTM 1 BAR (PCIC0_PTM1BAR)

PCIC0_PTM1BAR defines a space in PCI memory space mapped to PLB space (system memory or ROM).

Figure 21-36. PCI PTM 1 BAR Register (PCIC0_PTM1BAR)			
31:12	BA	Base Address These bits determine where in PCI memory address space this region is located.	Only corresponding bits in PCIL0_PTM1MS that are set to 1 are writable. Bits in PCIL0_PTM1MS that are set to 0 cause the corresponding Base Address register bits to be always 0. PCIL0_PTM1MS must be initialized by a PLB master before any PCI device is allowed to configure this register.
11:4	BAZ	Base Address Always Zero	BAZ = 0x00 because the minimum size of this range is 4KB.
3	PF	Prefetchable	PF = 1 to indicate that prefetching is allowed.
2:1	LT	Location Type	LT = 0b00 to indicate that the memory space can be located anywhere in the 32-bit address space.
0	MSI	Memory Space Indicator	MSI = 0 to indicate memory space, rather than I/O space.

21.6.3.15 PCI PTM 2 BAR (PCIC0_PTM2BAR)

PCIC0_PTM2BAR defines a second space in PCI memory space that is mapped to PLB space (system memory or ROM). Note that if PCIC0_PTM2BAR is disabled using PCIL0_PTM2MS, PCIC0_PTM2BAR cannot be written. Set PCIC0_PTM2BAR to 0 before disabling this bit. If disabled in this way, reads to PCIC0_PTM2BAR always return 0.

Figure 21-37. PCI PTM 2 BAR Register (PCIC0_PTM2BAR)			
31:12	BA	Base Address These bits determine where in PCI Memory address space this region is located.	Only corresponding bits in PCIL0_PTM2MS that are set to 1 are writable. Bits in PCIL0_PTM2MS that are set to 0 cause the corresponding Base Address register bits to be always 0. PCIL0_PTM2MS must be initialized by a PLB master before any PCI device can configure this register.
11:4	BAZ	Base Address Always Zero	BAZ = 0x00 because the minimum size of this range is 4KB.
3	PF	Prefetchable	PF = 1 to indicate that prefetching is allowed.

2:1	LT	Location Type	LT = 0b00 to indicate that the memory space can be located anywhere in the 32-bit address space.
0	MSI	Memory Space Indicator	MSI = 0 to indicate memory space, rather than I/O space.

21.6.3.16 PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)

PCIC0_SBSYSVID holds the vendor ID for a subsystem or add-in board.

Figure 21-38. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)

15:0		PCI Subsystem Vendor ID	

21.6.3.17 PCI Subsystem ID Register (PCIC0_SBSYSID)

PCIC0_SBSYSID holds the device ID of a subsystem or add-in board.

Figure 21-39. PCI Subsystem ID Register (PCIC0_SBSYSID)

15:0		PCI Subsystem ID	

21.6.3.18 PCI Capabilities Pointer (PCIC0_CAP)

PCIC0_CAP contains an 8-bit pointer in configuration space to the next capability. This data structure is indicated by PCIC0_STATUS[CL]. PCIC0_CAP points to the first item in the list of capabilities at address offset 0x58, which is the PCI power management capability structure.

Figure 21-40. PCI Capabilities Pointer (PCIC0_CAP)

7:0		PCI Capabilities Pointer	

21.6.3.19 PCI Interrupt Line Register (PCIC0_INTLN)

PCIC0_INTLN contains interrupt line routing information.

Figure 21-41. PCI Interrupt Line Register (PCIC0_INTLN)

7:0		PCI Interrupt Line	

21.6.3.20 PCI Interrupt Pin Register (PCIC0_INTPN)

PCIC0_INTPN specifies the PCI interrupt line that the device uses. The value 0x01 indicates INTA#.

Figure 21-42. PCI Interrupt Pin Register (PCIC0_INTPN)			
7:0		PCI Interrupt Pin	

21.6.3.21 PCI Minimum Grant Register (PCIC0_MINGNT)

PCIC0_MINGNT specifies the burst period length of a PCI device. PCIC0_MINGNT is read-only and returns 0x00 when read.

Figure 21-43. PCI Minimum Grant Register (PCIC0_MINGNT)			
7:0		PCI Minimum Grant	

21.6.3.22 PCI Maximum Latency Register (PCIC0_MAXLTNCY)

PCIC0_MAXLTNCY specifies how often a PCI device needs to access to the PCI bus. PCIC0_MAXLTNCY is read-only and returns 0x00 when read.

Figure 21-44. PCI Maximum Latency Register (PCIC0_MAXLTNCY)			
7:0		PCI Maximum Latency	

21.6.3.23 PCI Interrupt Control/Status Register (PCIC0_ICS)

A PLB master or a PCI master device may generate an interrupt to the PCI bus by writing a 1 to bit 0. Clearing this bit clears the interrupt. Bit 0 also reports the status of the interrupt. A value of 1 means that the interrupt is asserted, a value of 0 means that the interrupt is deasserted.

Figure 21-45. PCI Interrupt Control/Status Register			
7:1		Reserved	These bits return 0 when read.
0	API	Assert PCI interrupt	When software sets this bit, the PCI bridge asserts its Interrupt pin.

21.6.3.24 Error Enable Register (PCIC0_ERREN)

ERREN enables detection and reporting of various errors for the PCI bridge (see *Error Handling* on page 493).

Figure 21-46. PCI Error Enable Register (PCIC0_ERREN)			
7		Reserved	

6	TAEE	Target Abort Error Enable 0 Disabled 1 Enabled	While the PCI bridge is the PCI master, this bit enables the detection of a target abort as an error condition. If TAEE is enabled, the PCI bridge reports PLB bus errors.
5:4	MERE	PLB Bus Error Response Enable 00 No action is taken. 01 The PCI target should drive $\overline{\text{PCISErr}}$ on the PCI bus. 10 Target should target abort the offending read. 11 Indicates the PCI target should drive $\overline{\text{PCISErr}}$ and target abort.	MERE controls the response taken by the PCI bridge on the PCI bus (as the PCI target) when PLB bus errors are asserted to the PCI bridge PLB master. **Note:** Only reads can be target aborted. Modes 10 and 11 cannot be used in asynchronous mode.
3	MEDE	PLB Master Error Detection Enable 0 Disables detection of PLB master errors. 1 Enables detection of PLB master errors.	MEDE enables the detection of PLB bus errors when the PCI bridge is a PLB master.
2	MEAE	PLB Bus Error Assertion Enable 0 Disabled 1 Enabled	MEAE enables the reporting of a PLB bus error when the PCI bridge is a PLB slave.
1	WDPE	Write Data Parity $\overline{\text{PCISErr}}$ Enable 0 Disabled 1 Enabled.	The PCI bridge drives $\overline{\text{PCISErr}}$ when a data parity error is detected on a write cycle when the PCI bridge is the PCI target. PCIC0_CMD[SE] must also be 1.
0	MAEE	Master Abort Error Enable 0 Disabled 1 Enabled	MAEE enables the detection of a master abort as an error condition when the PCI bridge is the master. The PCI bridge drives Sl_MErr on the PLB bus in response to a master abort. If this bit is disabled, driving of Sl_Merr in response to master abort is masked.

21.6.3.25 Error Status Register (PCIC0_ERRSTS)

PCIC0_ERRSTS contains status for detected error conditions (see *Error Handling* on page 493). Bits in PCIC0_ERRSTS can be set to 1 only as the result of a system error. Writing a 1 to a PCIC0_ERRSTS bit clears the bit. Writing a 0 to a bit leaves that bit unchanged.

Figure 21-47. PCI Error Status Register (PCIC0_ERRSTS)

7:5		Reserved	
4	SARME	$\overline{\text{PCISErr}}$ Asserted on Received PLB Bus Error	Set when PCI bridge asserts $\overline{\text{PCISErr}}$ on the PCI bus in response to PCI bridge receiving a PLB bus error while PLB master.
3	MED	PLB Bus Error Detected Error detected	Set when a PLB bus error signal is asserted when PCI bridge is the PLB master. MED is set regardless of whether the PCI bridge is enabled to treat this as an error condition (the setting of MED is not maskable).
2	MEAE	PLB Bus Error Assertion Event An PCI bridge error, which can cause a PLB bus error, occurred.	Set when an error occurs that would cause PCI bridge (as PLB slave) to assert a PLB bus error signal. MEAE is set regardless of whether the PLB bus error assertion is enabled (the setting of MEAE is not maskable).
1	WDPE	$\overline{\text{PCISerr}}$ on Write Data Parity Error	Set when the PCI bridge drives $\overline{\text{PCISErr}}$ in response to a data parity error detected on a PCI write to PLB memory. $\overline{\text{PCIPErr}}$ is also driven.
0	PUR	PLB Unsupported Request	Set when the PCI bridge is a PLB slave and detects an unsupported request from a PLB master to an address range that PCI bridge decodes. The PCI bridge allows such requests to time out.

21.6.3.26 Bridge Options 1 Register (PCIC0_BRDGOPT1)

PCIC0_BRDGOPT1 controls various operating parameters of the PCI bridge. The parameters must be initialized before PCI masters access the PCI bridge.

Figure 21-48. PCI Bridge Options 1 Register (PCIC0_BRDGOPT1)

15:8	PMLTCR	PLB Master Latency Timer Count Register	PMLTCR contains the value used by the PLB master to load its latency timer. The granularity of this timer is 16 PLB cycles; therefore, the low-order bits of this register are read-only and are hard wired to 1.
7	PLESE	PLB Lock Error Status Enable 0 Slave error locking is disabled. 1 Slave error locking is enabled.	PLESE controls the handling of slave error locking.
6:5	PRP	PLB Request Priority 11 Highest 10 Next highest 01 Next highest 00 Lowest	PRP controls the request priority for PLB accesses.
4	PGMAE	PLB Guarded Memory Access Enable 0 Bridge PLB master memory accesses are unguarded. 1 Bridge PLB master memory accesses are guarded.	PGMAE controls whether PLB accesses are guarded or unguarded.
3	PAPM	PCI Arbiter Park Mode 0 The arbiter parks on requester 0 (the bridge PCI master). 1 The arbiter parks on the last master granted the bus.	PAPM defines how the internal PCI arbiter handles bus parking.
2:1	PTMRCI	PCI Target Memory Read Command Interpretation 00 Memory Read 01 Memory Read Line 10 Memory Read Multiples 11 Reserved	PTMRCI enables the PCI bridge to be forced to treat a PCI memory read as a memory read multiple, or as a memory read line, with respect to the burst size implied by the read commands. This is for masters that use memory read for multiple beat bursts.
0	APLRM	Atomic PLB Line Read Mode 0 1 PLB slave asserts AddrAck and begins its data tenure immediately after the PCI master receives the first read data word.	APLRM controls the behavior of the bridge PLB slave with respect to PLB line reads. APLRM must not be se t to 1 unless all PCI target devices can guarantee no disconnects for PLB line reads.

21.6.3.27 PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)

PCIC0_PLBBESR0 stores information about errors reported by the PCI bridge PLB slave. There are four groups of errors, one each for PLB masters 0–3. PCIC0_PLBBESR0[MxET] fields (x represents a particular PLB master ID) contain information about the type of error. PCI parity errors set PCIC0_PLBBESR0[MxET] to 0b001. Master and target aborts set PCIC0_PLBBESR0[MxET] to 0b101 (non-configured bank error). The PCIC0_PLBBESR0[MxRWS] fields show whether the transaction causing the error was a read or write.

Each error field can be locked by PCIC0_PLBBESR0[MxFL], which is set by a PLB lock error signal to the bridge PLB slave. If the PCIC0_PLBBESR0[MxFL] field associated with a master is 0, the PLB lock error signal is driven high to the bridge PLB slave, and an error associated with that master occurs. The error is then reported, and PCIC0_PLBBESR0[MxFL] is set. Subsequent errors do not set PCIC0_PLBBESR0 fields for that master until

PCIC0_PLBBESR0[MxFL] is cleared. If PCIC0_PLBBESR0[MxFL] = 0, and the PLB lock error signal is low, the error is reported, and PCIC0_PLBBESR0[MxFL] is not set. Additional errors are also reported. Only software can clear PCIC0_PLBBESR0[MxFL].

Writing 1 to a PCIC0_PLBBESR0 field clears the field.

Figure 21-49. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)

31:29	M0ET	Master 0 Error Type No Error Parity Error Reserved Reserved Reserved Non-configured Bank Error Reserved Reserved	
28	M0RWS	Master 0 Read/Write Status Error operation was a write Error operation was a read	
27	M0FL	Master 0 PCIC0_PLBBESR0 Field Lock PCIC0_PLBB ESR0 unlocked PCIC0_PLBB ESR0 locked	
26	M0AL	Master 0 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 0 PCIC0_PLBBEAR locked by Master 0	
25:23	M1ET	Master 1 Error Type	See PCIC0_PLBBESR0[M0ET]
22	M1RWS	Master 1 Read/Write Status Error operation was a write Error operation was a read	
21	M1FL	Master 1 PCIC0_PLBBESR0 Field Lock PCIC0_PLBB ESR0 unlocked PCIC0_PLBB ESR0 locked	
20	M1AL	Master 1 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 1 PCIC0_PLBBEAR locked by Master 1	
19:17	M2ET	Master 2 Error Type	See PCIC0_PLBBESR0[M0ET]
16	M2RWS	Master 2 Read/Write Status Error operation was a write Error operation was a read	
15	M2FL	Master 2 PCIC0_PLBBESR0 Field Lock PCIC0_PLBB ESR0 unlocked PCIC0_PLBB ESR0 locked	
14	M2AL	Master 2 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 2 PCIC0_PLBBEAR locked by Master 2	
13:11	M3ET	Master 3 Error Type	See PCIC0_PLBBESR0[M0ET]
10	M3RWS	Master 3 Read/Write Status Error operation was a write Error operation was a read	

9	M3FL	Master 3 PCIC0_PLBBESR0 Field Lock PCIC0_PLBBESR0 unlocked PCIC0_PLBBESR0 locked	
8	M3AL	Master 3 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR Unlocked by Master 2 PCIC0_PLBBEAR Locked by Master 2	
7:0		Reserved	

21.6.3.28 PLB Slave Error Syndrome Register 1 (PCIC0_PLBBESR1)

PCIC0_PLBBESR1 stores information about errors reported by the bridge PLB slave. There are four groups of errors, one each for PLB masters 4–7. See *PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)* on page 487 for additional information about the fields of this register.

Only software can clear PCIC0_PLBBESR1[MxFL]. The PCIC0_PLBBESR1[MxAL] fields control the and PCIC0_PLBBESR0 and PCIC0_PLBBESR1 in the same way. Writing a 1 to a field of the PCIC0_PLBBESRx clears the bit.

Figure 21-50. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1)

31:29	M4ET	Master 4 Error Type No Error Parity Error Reserved Reserved Reserved Non-configured Bank Error Reserved Reserved	
28	M4RWS	Master 4 Read/Write Status Write error operation Read error operation	
27	M4FL	Master 4 PCIC0_PLBBESR1 Field Lock PCIC0_PLBBESR1 unlocked PCIC0_PLBBESR1 locked	
26	M4AL	Master 4 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 4 PCIC0_PLBBEAR locked by Master 4	
25:23	M5ET	Master 5 Error Type	See PCIC0_PLBBESR1[M4ET]
22	M5RWS	Master 5 Read/Write Status Write error operation Read error operation	
21	M5FL	Master 5 PCIC0_PLBBESR1 Field Lock PCIC0_PLBBESR1 Unlocked PCIC0_PLBBESR1 Locked	
20	M5AL	Master 5 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 5 PCIC0_PLBBEAR locked by Master 5	
19:17	M6ET	Master 6 Error Type	See PCIC0_PLBBESR1[M4ET]

16	M6RWS	Master 6 Read/Write Status Write error operation Read error operation	
15	M6FL	Master 6 PCIC0_PLBBESR1 Field Lock PCIC0_PLBBESR1 unlocked PCIC0_PLBBESR1 locked	
14	M6AL	Master 6 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 6 PCIC0_PLBBEAR locked by Master 6	
13:11	M7ET	Master 7 Error Type	See PCIC0_PLBBESR1[M4ET]
10	M7RWS	Master 7 Read/Write Status Write error operation Read error operation	
9	M7FL	Master 7 PCIC0_PLBB ESR1 Field Lock PCIC0_PLBBESR1 unlocked PCIC0_PLBBESR1 locked	
8	M7AL	Master 7 PCIC0_PLBBEAR Address Lock PCIC0_PLBBEAR unlocked by Master 7 PCIC0_PLBBEAR locked by Master 7	
7:0		Reserved	

21.6.3.29 PLB Slave Error Address Register (PCIC0_PLBBEAR)

PCIC0_PLBBEAR contains addresses associated with errors, as indicated by the PLB slave asserting Sl_MErr for transactions initiated by the PCI bridge on the PCI bus. PCIC0_PLBBEAR is read-only.

Figure 21-51. PLB Slave Error Address Register (PCIC0_PLBBEAR)

31:0		PLB Slave Error Address

21.6.3.30 Capability Identifier (PCIC0_CAPID)

When PCIC0_CAPID contains 0x01, the PCI bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.

Figure 21-52. PCI Capability Identifier (PCIC0_CAPID)

7:0		PCI Capability Identifier	

21.6.3.31 Next Item Pointer (PCIC0_NEXTIPTR)

PCIC0_NEXTIPTR describes the location of the next item in the capability list of the function. PCIC0_NEXTIPTR is set to 0x00 to indicate that this is the last item on the capability list.

Figure 21-53. PCI Next Item Pointer (PCIC0_NEXTIPTR)

7:0		PCI Next Item Pointer	

21.6.3.32 Power Management Capabilities (PCIC0_PMC)

PCIC0_PMC provides information about the capabilities of the function related to power management. A value of 0x0202 indicates no specific capabilities.

Figure 21-54. Power Management Capabilities Register (PCIC0_PMC)

15:11	PMES	PME Support	The PCI bridge does not support PME#; therefore, PMES is hard wired to 0b00000.
10	D2S	D2 Support Determines if the D2 power management state is supported.	The PCI bridge does not support the D2 power management state; therefore, D2S is hard wired to 0.
9	D1S	D1 Support Determines if the D1 power management state is supported.	The PCI bridge supports the D1 power management state; therefore, D1S is hard wired to 1.
8:6	AUXCUR	Auxiliary Current Support	The PCI bridge does not support Aux_Current; therefore, AUXCUR is hard wired to 0b000.
5	DSI	Device Specific Initialization after reset	This bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it.
4		Reserved	Always read as 0.
3	PMECLK		This bit is hard wired to 0 indicating that the function does not support PME# generation in any state.
2:0	VERS		Returns 0b010 on reads, indicating that PMC complies with Revision 1.1 of *PCI Power Management Interface Specification*.

21.6.3.33 Power Management Control/Status Register (PCIC0_PMCSR)

PCIC0_PMCSR is used to manage the PCI power management state and to enable and monitor PMEs.

Figure 21-55. Power Management Control/Status Register (PCIC0_PMCSR)

15	PMEST		The PCI bridge does not support PME#; therefore, PMEST is hard wired to 0.
14:13	DSCAL		The PCI bridge does not support data register; therefore, DSCAL is hard wired to 0b00.
12:9	DSEL		The PCI bridge does not support a data register; therefore, DSEL is hard wired to 0b0000.
8	PMEEN		The PCI bridge does not support PME generation; therefore, PMEEN is hard wired to 0.
7:2		Reserved	Returns 0 when read.
1:0	PSTAT	Determine the current power state of a function and sets the function into a new power state. D0 D1 D2 D3 Hot	If software attempts to write a value for an unsupported power state to PSTAT, its value does not change. Writing this field may change PCIC0_PMSCRR.

21.6.3.34 PMCSR PCI-to-PCI Bridge Support Extensions (PCIC0_PMCSRBSE)

PCIC0_PMCSRBSE is required for all PCI-to-PCI bridges. The PCI bridge in not a PCI-to-PCI bridge; therefore, it returns 0 when this register is read.

Figure 21-56. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBSE)			
7:0		PCI to PCI Bridge Support Extensions	

21.6.3.35 PCI Data Register (PCIC0_DATA)

PCIC0_DATA is an optional register that provides a mechanism for the function to report state dependent operating data such as power consumed or heat dissipation. The PCI bridge does not implement this register; therefore, it returns 0 when this register is read.

Figure 21-57. PCI Data Register (PCIC0_DATA)			
7:0		PCI Data	

21.6.3.36 Bridge Options 2 Register (PCIC0_BRDGOPT2)

PCIC0_BRDGOPT2 controls various operating parameters of the PCI bridge.

Figure 21-58. PCI Bridge Options 2 Register (PCIC0_BRDGOPT2)			
15:14		Reserved	
13	EWPCI	External Write to PCI Command Interrupt 0 No write to PCIC0_CMD has occurred. 1 External PCI master has written to PCIC0_CMD.	Software can set or clear this bit. Setting this bit also causes UIC0_SR[PCIIS] to be set.
12	DPR	Drive PCI Reset 0 Normal operation 1 Causes $\overline{\text{PCIReset}}$ pin to be asserted.	Software that asserts this bit must leave it asserted long enough to guarantee the PCI pulse width requirements. DPR does not reset PLB bus interface registers or PCI bridge registers. $\overline{\text{PCIReset}}$ is also asserted when the PCI bridge is reset.
11:8	PSTLTD	Subsequent Target Latency Timer Duration Specifies the number of PCI clocks that a PCI master burst can be held in a wait state before a target disconnect is initiated.	Only set on reads. In synchronous mode, PSTLTD equals the maximum number of PCI clocks to disconnect. In asynchronous mode, PSTLTD plus 3 equals the maximum number of PCI clocks to disconnect. The asynchronous value must be 2 or less.
7:3		Reserved	
2	PDTD	PCI Discard Timer Disable 0 Disabled 1 Enabled	When enabled, the PCI bridge never discards delayed read data.
1		Reserved	
0	HCE	Host Configuration Enable 0 Disabled 1 Enabled	HCE controls host PCI access to the PCI bridge configuration registers. All host attempts to access the PCI bridge PCI configuration registers are retried. This give the local CPU (PLB master) time to initialize them before the host sees them.

In synchronous mode, the PCI subsequent target latency timer duration equals the maximum number of PCI clocks to disconnect. In asynchronous mode, PCI subsequent target latency timer duration plus 3 equals the maximum number of PCI clocks to disconnect. The asynchronous value must be 2 or less.

21.6.3.37 Power Management State Change Request Register (PCIC0_PMSCRR)

PCIC0_PMSCRR provides a method of informing the local processor of a power management state change request and prevents the completion of the write to PCIC0_PMCSR until the local processor indicates it is ready for the state change. All writes to PCIC0_PMCSR are retried until the local processor sets PCIC0_PMSCRR[APW] = 1. PCIC0_PMSCRR is used with the registers in the capability structure for power management. Descriptions of each bit are shown in *Figure 21-59*.

Figure 21-59. Power Management State Change Request Register (PCIC0_PMSCRR)

7:5		Reserved	Always read as 0.
4	APW	Accept PCIC0_PMCSR Writes Always 1 if DWE is 0.	The local processor sets APW when the local processor is ready to change the power management state. APW is cleared when the host configuration writes to the PCIC0_PMCSR register is accepted. The local processor can write 0 to APW.
3	SCR	State Change Request	The PCI bridge sets SCR when a host writes PCIC0_PMCSR to request a power management state change. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear SCR and set APW = 1 when the local processor is ready to change the state. After SCR is cleared, new requests are not detected until the outstanding delayed write is accepted. The local processor can set SCR = 1. Note that any host side write to any byte (0x5C–0x5F) is considered a power state change request.
2:1	REQST	Request State	Indicates the new power management state requested by a delayed host write to PCIC0_PMCSR. This field is read-only from the PLB side.
0	DWE	Delayed Write Enable Immediate write Delayed write	When DWE is set to 1, any configuration write to the PCIC0_PMCSR is completed as a delayed write. All writes to PCIC0_PMCSR are retried until the local processor sets the "Accept PCIC0_PMCSR Write bit" (bit 4). When 0, any configuration write to the PCIC0_PMCSR is completed immediately. DWE is a don't care if a host write to PCIC0_PMCSR requests a state change from D3hot to D0.

21.7 Error Handling

The PCI bridge detects and reports several types of errors, which are reported to the PLB or the PCI. Status information is saved in the configuration registers to enable error type determination.

All errors are associated with either a cycle on the PLB or a cycle on the PCI bus.

Each error that can be detected is associated with a mask. If the mask is set, detection of that error condition is disabled. There are also masks for the PCISErr, PCIPErr, and PLB bus error signals that prevent reporting of any error by these signals. The masks do not prevent error detection.

The error types are as follows:

- PLB unsupported transfer type
- PCI master abort generated (while PCI master)
- PCI target abort received (while PCI master)
- PCI target data bus parity error detection
- PCI master data bus parity error detection
- PCI target address parity error detection
- PLB bus error detection

The following sections describe in detail how these errors are generated, what actions are taken for each, and how to reset a given error.

21.7.1 PLB Unsupported Transfer Type

This error occurs when the bridge PLB slave encounters an unsupported PLB transfer type. *Table 21-11* outlines transfers not supported by the bridge PLB slave.

Table 21-11. PLB Unsupported Transfer Types

PLB Transaction	PCI Address Space
4- and 8-word line read/write	Nonmemory
16-word line read/write	Any
Burst	Nonmemory

Upon detection of this error, the bridge sets PCIC0_ERRSTS[PUR] = 1.

21.7.2 PCI Master Abort

This error is generated by the bridge PCI master when no target responds with $\overline{\text{PCIDevSel}}$ within the required time-out window and error detection is enabled. The bridge PLB slave may assert a PLB bus error signal on the PLB in response to this error, as explained below.

Two masks are associated with a PCI master abort. PCIC0_ERREN[MAEE] masks error reporting. If the error is detected, a PLB bus error signal is asserted if PCIC0_ERREN[MAEE] = 1. For reads, the bridge PLB slave still completes the transfer on the PLB, but drives 1s on the read data bus and the appropriate PLB bus error signal for each data beat. For posted writes, a PLB bus error is asserted for 1 cycle, asynchronously to the corresponding write data beat on the PLB. For connected writes, a PLB bus error signal is asserted with the data transfer, and the data is discarded. If PCIC0_ERREN[MAEE] = 0, error reporting is masked. No PLB bus error signal is asserted, regardless of the setting of PCIC0_ERREN[MEAE].

The following status bits are set:

1. If a master abort is signalled, PCIC0_STATUS[RMA] = 1. Setting of this field is non-maskable. Writing a 1 to PCIC0_STATUS[RMA] resets the field.
2. If master abort is detected as an error, PCIC0_ERRS[MEAE] is set to 1 to indicate an event that would cause a PLB bus error to be asserted by the bridge PLB slave, regardless of the setting of PCIC0_ERREN[MEAE]. This field can be reset by writing a 1 to PCIC0_ERREN[MEAE].
3. PCIC0_PLBBEAR and PCIC0_PLBBESRx are updated as follows:

The address of the aborted request is saved in PCIC0_PLBBEAR if all PCIC0_PLBBESRx[MxAL] = 0 (PCIC0_PLBBEAR is unlocked). If all PCIC0_PLBBESRx[MxFL] = 0, PCIC0_PLBBESRx[MxET] = 0b101 to indicate a non-configured bank error; and PCIC0_PLBBESRx[MxRWS] is set to 0 on a write, 1 on a read. If PCIC0_ERREN[MAEE] = 0 or PCIC0_ERREN[MEAE] = 0, no PCIC0_PLBBEAR or PCIC0_PLBBESRx update is performed.

21.7.3 Bridge PCI Master Receives Target Abort While PCI Bus Master

This error is generated when the bridge PCI master receives a target abort while mastering a cycle on the PCI bus. Upon detection of this error, the bridge PLB slave may assert a PLB bus error signal on the PLB in response to this error, as explained below.

Two masks are associated with a target abort. PCIC0_ERREN[TAEE] masks error reporting. If the error is detected, a PLB bus error signal is asserted if PCIC0_ERREN[TAEE] = 1. For reads, the bridge PLB slave still completes the transfer on the PLB and drives the appropriate PLB bus error line for each data beat (note that for line reads, a PLB bus error signal is asserted for all data beats). For posted writes, if PCIC0_ERREN[TAEE] = 1, the bridge PLB slave asserts a PLB bus error for 1 cycle, asynchronously to the corresponding write data beat on the PLB. For connected writes, a PLB bus error signal is asserted with the data transfer, and the data is discarded. If PCIC0_ERREN[TAEE] = 0, error reporting is masked No PLB bus error signal is asserted, regardless of the setting of PCIC0_ERREN[MEAE]. If prefetching is occurring when a target abort is received, data preceding the target abort is kept in a prefetch buffer.

The following status bits are set:

1. If a target abort is received, PCIC0_STATUS[RTA] = 1. Setting this field is non-maskable. Writing a 1 to PCIC0_STATUS[RTA] clears the field.
2. If a target abort is detected as an error, PCIC0_ERRSTS[MEAE] = 1 to indicate an event that would cause the bridge PLB slave to assert a PLB bus error signal, regardless of the setting of PCIC0_ERREN[MEAE]. Writing a 1 to PCIC0_ERRSTS[MEAE] resets the field.
3. PCIC0_PLBBEAR and PCIC0_PLBBESRx are updated as follows:

   The address of the aborted request is saved in PCIC0_PLBBEAR if all PCIC0_PLBBESRx[MxAL] = 1 (PCIC0_PLBBEAR is unlocked). If all PCIC0_PLBBESRx[MxFL] = 1, PCIC0_PLBBESRx[MxET] = 0b101 to indicate a nonconfigured bank error, and PCIC0_PLBBESRx[MxRWS] is set to 0 on a write, or to 1 on a read. If PCIC0_ERREN[MAEE] = 0 or PCIC0_ERREN[MEAE] = 0, no PCIC0_PLBBEAR or PCIC0_PLBBESRx update is performed.

21.7.4 PCI Target Data Bus Parity Error Detection

This error is generated when the bridge PCI target detects a data bus parity error on write data from a PCI master doing a write cycle to PLB memory. PCI uses even parity.

Setting PCIC0_CMD[PER] = 0 masks this error.

The following status bits are set:

1. PCIC0_STATUS[DEPE] = 1 to indicate a PCI bus parity error. Setting this field is non-maskable. Writing a 1 to PCIC0_STATUS[DEPE] clears the field.
2. PCIC0_STATUS[SSE] = 1 if PCIC0_ERREN[WDPE] = 1. Writing a 1 to PCIC0_STATUS[SSE] clears the field.
3. PCIC0_ERRSTS[WDPE] = 1 if PCIC0_ERREN[WDPE] = 1. Writing a 1 to PCIC0_ERRSTS[WDPE clears the field.

21.7.5 PCI Master Data Bus Parity Error Detection

This error is generated when a data bus parity error is detected on the PCI bus during a cycle mastered by the bridge PCI master. The bridge PCI master checks parity on read cycles and samples PCIPErr on write cycles. The bridge PCI master may assert PCIPErr if the master detects a parity error on a read. PCI uses even parity.

Setting PCIC0_CMD[PER] = 0 masks this error. PCIC0_STATUS[DEPE] = 1. If a parity error is detected, writing a 1 to PCIC0_STATUS[DEPE] = 1 clears the field.

If PCIC0_ERREN[MEAE] = 1 and the error is detected as described, the PLB slave asserts a PLB error signal on the PLB in response to the error. For reads, a PLB bus error is asserted for each data beat in which bad parity was detected. For writes, a PLB bus error is asserted for each data beat in which bad parity was detected, but asynchronously to the actual transfer of write data on the PLB.
The following status bits are set:

1. PCIC0_STATUS[DEPE] = 1 if the bridge PCI master detects bad parity on read data, regardless of the state of PCIC0_CMD[PER]. Writing a 1 to PCIC0_STATUS[DEPE] clears the field.
2. If a data bus parity error is detected as an error, PCIC0_ERRSTS[MEAE] = 1 to indicate an event that would cause a PLB bus error signal to be asserted by the bridge PLB slave, regardless of the state of PCIC0_ERREN[MEAE]. Writing a 1 to PCIC0_ERRSTS[MEAE] = 1 clears the field.
3. PCIC0_PLBBEAR and the PCIC0_PLBBESRx are updated as follows:

   The address of the PCI transaction where parity errors occurred is saved in the PCIC0_PLBBEAR. PCIC0_PLBBEAR is set if all PCIC0_PLBBESRx[MxAL] = 1 (PCIC0_PLBBEAR is unlocked). If PCIC0_PLBBESRx[MxFL] = 1, PCIC0_PLBBESRx[MxET] = 0b001 to indicate a parity error, and PCIC0_PLBBESRx[MxRWS] is set to 0 on a write, 1 on a read. If PCIC0_CMD[PER] = 0 or PCIC0_ERRSTS[MEAE] = 0, no PCIC0_PLBBEAR or PCIC0_PLBBESRx update is performed.

Note: For clock ratios greater than 2:1 (independent of asynchronous/synchronous mode), the PCI bridge detects errors but does not assert a PLB bus error signal or log the error in PCIC0_PLBBEAR, PCIC0_PLBBESRx, and PCIC0_ERRSTS[MEAE].

21.7.6 PCI Address Bus Parity Error While PCI Target

This error occurs when a PCI address bus parity error is detected during the address phase of a cycle in which the bridge is the PCI target. PCI uses even parity.

Setting PCIC0_CMD[PER] = 0 masks this error. This error does not have an explicit status bit, however the following actions are taken:

1. PCIC0_STATUS[SSE] = 1 to indicate assertion of $\overline{\text{PCISErr}}$, if the mask at PCIC0_CMD[SE] = 1. Writing a 1 to PCIC0_STATUS[SSE] clears the field.
2. PCIC0_STATUS[DEPE] = 1 to indicate a PCI bus parity error, regardless of the state of PCIC0_CMD[PER]. PCIC0_STATUS[DEPE] = 1 when any type of PCI parity error is detected. Writing a 1 to PCIC0_STATUS[DEPE] clears the field.

21.7.7 PLB Master Bus Error Detection

This error occurs when the bridge PLB master detects a PLB bus error. If the bridge PLB master receives a PLB bus error while mastering a read, the master associates the error with the currently executing read. If the master receives a PLB bus error while mastering a write or while idle, the master associates the error with a write.

PLB bus error detection is masked by PCIC0_ERREN[MEDE] = 0. If PCIC0_ERREN[MEDE] = 1, PLB bus error detection is enabled.

PCIC0_ERREN[MERE] controls the response of the bridge PCI target to PLB bus error detection. If PCIC0_ERREN[MERE] = 10 or 11, the bridge PCI target will execute a target abort. If PCIC0_ERREN[MERE] = 01 or 11, the bridge PCI target asserts PCISErr and allows the transaction to continue. If PCIC0_ERREN[MERE] = 11, the bridge PCI target both target aborts and asserts PCISErr.

The following status bits are set:

1. If the bridge PCI target executes a target abort, PCIC0_STATUS[STA] = 1. The setting of PCIC0_STATUS[STA] in such an event is non-maskable. Writing a 1 to PCIC0_STATUS[STA] clears the field.
2. f the bridge PCI target asserts PCISErr, PCIC0_STATUS[SSE] = 1. The setting of PCIC0_STATUS[SSE] is non-maskable. Writing a 1 to PCIC0_STATUS[SSE] clears the field.
3. If the bridge PCI target asserts PCISErr, PCIC0_ERRSTS[SARME] = 1 to indicate that the bridge PCI target asserted PCISErr in response to a received PLB bus error signal. The setting of PCIC0_ERRSTS[SARME] is non-maskable. Writing a 1 to PCIC0_ERRSTS[SARME] clears the field.
4. PCIC0_ERRSTS[MED] = 1 to indicate that the bridge PLB master received a PLB bus error signal. Setting of PCIC0_ERRSTS[MED] is non-maskable. Writing a 1 to PCIC0_ERRSTS[MED] clears the field.

21.8 PCI Power Management Interface

The PCI bridge supports *PCI Power Management Interface Specification* Revision 1.1 (PCI-PM).

21.8.1 Capabilities and Power Management Status and Control Registers

The PCI bridge has a capabilities structure in the PCI configuration space that indicates that the PCI bridge core is PCI Power Management capable. The capabilities structure includes the following registers:

- PCIC0_CAPID, value 0x01, indicates Power Management
- PCIC0_NEXTIPTR, points to next capabilities structure
- PCIC0_PMC, value 0x0202, indicates no specific capabilities
- PCIC0_PMCSR, indicates hold the current power state
- PCIC0_PMCSR_BSE, value 0x00, unused in PCI-to-PCI bridge
- PCIC0_Data, value 0x00, not used
- See *PCI Configuration Registers* on page 477 for details.

21.8.2 Power State Control

The current power management state is reported by reading PCIC0_PMCSR. The PCI bridge supports states D0, D1, D3hot, and D3cold. State D2 is not supported. When the state is not D0, the PCI bridge is masked from being a master or a memory or I/O target on the PCI bus. The PCI bridge can still be a config target. Thus, accesses claimed by the PCI bridge when in state D0 are no longer claimed, resulting in master aborts on the PLB or PCI if such an access is attempted. Note that this mask is independent of the state of the PCI Command register.

21.8.3 Changing Power States

The PCI bridge has two registers that control changing the power state. The host requests a change in the power state by writing to the PCIC0_PMCSR. The other register is PCIC0_PMSCRR, which provides a method of informing the local processor of a state change request and of preventing completion of the write to the PCIC0_PMCSR until the local processor indicates that it is ready for the state change.

Power state changes are handled as follows:

- If a host write to PCIC0_PMCSR requests an unsupported state change (such as a change to D2), the write is accepted but is ignored (no state change occurs).
- If a host write to PCIC0_PMCSR requests a change from D3hot to D0, the write is accepted. Then, the PCI bridge asserts the power management reset signal, which causes the entire SOC to be reset.

Note: The PCI bridge assumes that any requested state change from D3hot is always to D0.

- All other change requests are handled with the following sequence:
  1. The host requests a new power state by a PCI write to the PCIC0_PMCSR.
  2. The host PCI write is retried (unless PCIC0_PMSCRR[DWE] = 0).
  3. The host PCI write (retried or not) sets PCIC0_PMSCRR[SCR] = 1, which drives an interrupt to the local processor.
  4. The local processor recognizes the interrupt. The local processor checks PCIC0_PMSCRR[SCR, REQST] to determine the nature of the request.
  5. The local processor proceeds to power down the subsystem if the requested state is valid.
  6. When the subsystem has been powered down and is ready to change state, the local processor clears the PCIC0_PMSCRR[SCR] and sets PCIC0_PMSCRR[APW] = 1.
  7. When the host PCI write reoccurs:
     It is accepted.
     PCIC0_PMSCR is updated (only if the transition is valid).
     PCIC0_PMSCRR[APW] = 0, unless PCIC0_PMSCRR[DWE] = 0, in which case
     PCIC0_PMSCRR[APW] = 1 always.
     The PCI bridge enters a new power state.

The PCI bridge operates with the clock power management (CPM) logic to enable the bridge to be put into sleep mode under control of software. See *Clock and Power Management* on page 291 for discussion of the CPM function.

21.9 PCI Bridge Reset and Initialization

The following sections discuss resetting and initializing the PCI bridge.

21.9.1 Address Map Initialization

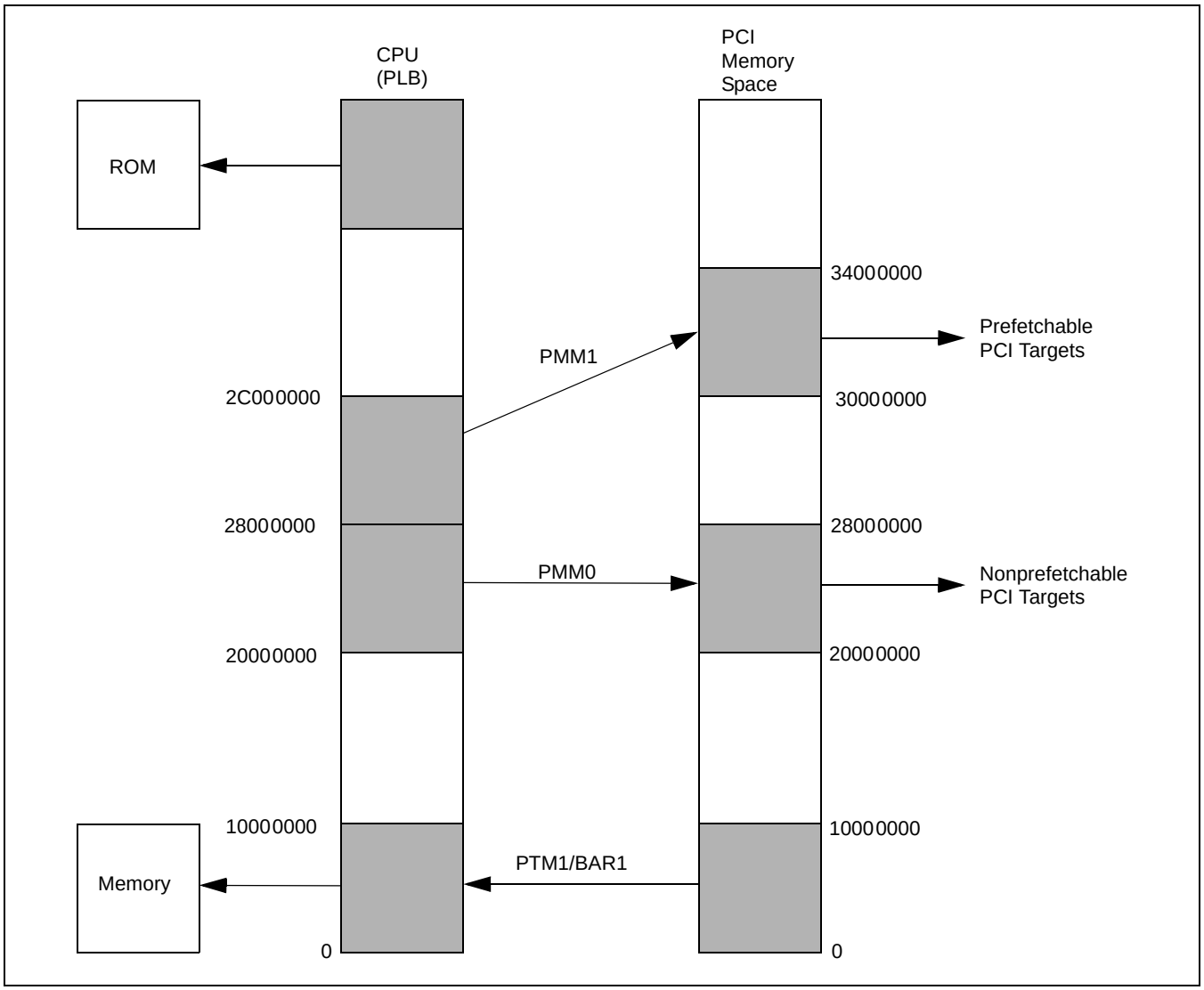
When the PCI bridge is the PCI master, it can generate memory, I/O, configuration, interrupt acknowledge, and special cycles. The method of cycle generation, and the associated address ranges, are fixed, except for memory cycles. PCI memory cycles are generated when the PCI bridge detects a cycle in one of three specified PLB address ranges. The sizes and address spaces of these ranges are specified using the PMM registers. Also, the address of the resulting PCI memory cycle can be an offset from the PLB address (address translation occurs). This translation is also specified in the PMM registers. The PMM registers *do not* default to usable values following reset; they must be initialized before attempting to generate PCI memory cycles.

When the PCI bridge is a target on the PCI bus, the PCI bridge can respond to memory cycles. The memory cycle address ranges that the PCI bridge responds to are specified in PCIC0_PTM1BAR and PCIC0_PTM2BAR. These registers are typically initialized as part of the standard PCI initialization process.

Figure 21-60 shows the desired address map. System memory resides from 0x0000 0000–0x0FFF FFFF in the CPU/PLB address space, which is accessible from the PCI in the same address space (PCI bridge as a memory target) as defined by PTM1/BAR1. PTM2/BAR2 is disabled in this example. The CPU/PLB master has two spaces

in which to access PCI Memory space. Range 0 is 0x20000000 to 0x27FFFFFF and is mapped to the same address on the PCI bus, and is nonprefetchable. Range 1 is 0x28000000 to 0x2BFFFFFF, and is translated to address range 0x30000000 to 0x33FFFFFF of PCI memory space. Range 2 is disabled.

Figure 21-60. Example Address Map



The following register values provide the address map shown in *Figure 21-60*:

Table 21-12. Address Map Register Values

Register Name	Value	Comments
PCIL0_PMM0LA	0x20000000	
PCIL0_PMM0MA	0xF8000001	128MB; enabled; read prefetching not allowed.
PCIL0_PMM0PCILA	0x20000000	
PCIL0_PMM0PCIHA	0x00000000	
PCIL0_PMM1LA	0x28000000	
PCIL0_PMM1MA	0xFC000003	64MB; enabled; prefetching allowed.

Table 21-12. Address Map Register Values (continued)

Register Name	Value	Comments
PCIL0_PMM1PCILA	0x30000000	
PCIL0_PMM1PCIHA	0x00000000	
PCIL0_PMM2LA	0x00000000	
PCIL0_PMM2MA	0x00000000	Not enabled.
PCIL0_PMM2PCILA	0x00000000	
PCIL0_PMM2PCIHA	0x00000000	
PCIL0_PTM1MSr	0xF0000001	256MB; enabled.
PCIL0_PTM1LA	0x00000000	
PCIL0_PTM2MS	0x00000000	Not enabled.
PCIL0_PTM2LA	0x00000000	
PCIC0_PTM1BAR	0x00000008	PCI memory space; address decode starts at PCI address 0x0000 0000.
PCIC0_PTM2BAR	0x00000000	

21.9.2 Other Configuration Register Initialization

Additional register initialization is required, as follows:

- Error handling is initially disabled (error detection is masked). If error handling is to be enabled, PCIC0_ERREN must be initialized appropriately.
- PCIC0_BRDGOPT1 contains options for controlling the PLB. Its default values can be used.
- PCIC0_BRDGOPT2 contains options for controlling the PCI bus. Its default values assume that the PCI is run synchronously to the PLB, and that the PCI bridge is the primary host bridge. If the PCI bridge is used otherwise, the values must be changed accordingly.

Note: PCI devices that are targets and support delayed reads may be attached to the PCI bus. To ensure that the PPC440EPx/GRx does not deadlock when accessing such devices, the PLB priority of the PCI and of all the PLB masters that access PCI space must be set to the same value, and the arbitration mode within the PLB arbiter must be set to fair mode. Since the processor data cache unit has dynamic PLB priority, the higher priority value of the DCU must be the same as the PCI. For additional information on setting PLB priorities see *PLB Master and Slave Assignments* on page 58.

21.9.3 Target Bridge Initialization

The PCI bridge can also respond as a configuration target; however, the PCI bridge only responds as a configuration target when the PCIIDSel pin is attached, rather than pulled inactive. Note that if the size and local address of these ranges are not strapped to desired values at reset, the local CPU must specify them by setting the PTM Memory Size and Local Address registers before initializing PCIC0_PTM1BAR and PCIC0_PTM2BAR.

The local CPU must update the following registers (if the default value is not suitable or they were not strapped to appropriate values at reset) before setting the Host Config Enable bit:

- The address map registers (see *Address Map Initialization* on page 498)
- PCIC0_VENDID
- PCIC0_DEVID

- PCIC0_REVID
- PCIC0_CLS
- PCIC0_SBSYSID
- PCIC0_SBSYSVID

21.9.4 Local Processor Boot from PCI Memory

The PCI bridge has a mode that enables a PLB master to access a PCI memory range without initial configuration cycles. This mode is enabled when CPC0_ = 1. System designers can use this mode to enable a processor to access a boot ROM in PCI memory space.

The PCI bridge comes out of reset with PMM0 enabled and programmed for the address range, 0xFFFE0000–0xFFFFFFFF. Also, PCIC0_CMD[ME] = 1 after reset. Note that enabling PCI boot mode does not prevent subsequent updates to the PMM0 registers.

Note: The PPC440EPx/GRx allows booting from PCI memory. See *Bootstrap Controller* on page 187 for more information.

21.9.5 Type 0 Configuration Cycles for Other Devices

Twenty-one devices can be accessed using the PCIIDSel mechanism. The PCI master asserts 1 bit of AD(31:11) for type 0 configuration cycles based on the value in the Device Number field. The mapping is as follows:

- If device number is 1, AD(11) is asserted
- If device number is 2, AD(12) is asserted
- .
- .
- .
- If device number is 21, AD(31) is asserted

If device number contains a value of 22–31, no bit of AD(31:11) is asserted.

21.10 Timing Diagrams

This section contains timing diagrams of several different PCI bridge operations. The following paragraphs describe each diagram in detail. Each description assumes basic knowledge of PCI and PLB protocols.

Each operation is shown in both synchronous and asynchronous modes. The PCI is clocked at 33 MHz in synchronous mode and 66 MHz in asynchronous mode. The PLB is clocked at 100 MHz in all cases.

The SDRAM uses a 32-bit, PC100 memory interface configured for $\overline{\text{CAS}}$ latency of 2, command leadoff of 2, and RAS to CAS delay (T_{rcd}) of 2. All memory accesses are page idle, unless indicated otherwise.

Note: The PLB signals shown in the following timing diagrams are not externally accessible. They are included for information purposes and as an aid to understanding the PCI operations. For more information on these signals, refer to *Processor Local Bus Architecture Specifications*.

21.10.1 PCI Timing Diagram Descriptions

The following sections briefly describe each of the timing diagrams that follow the descriptions. Each description covers both the asynchronous and synchronous clocking modes for that operation. The timing diagrams then follow with all of the asynchronous diagrams grouped together followed by all of the synchronous diagrams grouped together.

21.10.1.1 PCI Master Burst Read From SDRAM

Figure 21-61 PCI Master Burst Read From SDRAM on page 503 (asynchronous) and *Figure 21-83 PCI Master Burst Read From SDRAM* on page 525 (synchronous) show a PCI Master executing a 128-byte Read Multiple from SDRAM. PCI bridge retries the initial request and performs a delayed read. PCI bridge executes a variable-length, doubleword read burst on PLB to SDRAM, filling its 96-byte read buffer. The read is a page hit. When the PCI master re-requests its read, PCI bridge begins bursting out of its read buffer, while continuing to prefetch from SDRAM.

21.10.1.2 PCI Master Burst Write To SDRAM

Figure 21-65 PCI Master Burst Write To SDRAM on page 507 and *Figure 21-90 PCI Master Burst Write To SDRAM* on page 532 show a PCI Master executing a 128-byte Write to SDRAM. PCI bridge accepts several beats of data into its 64-byte write buffer before executing variable-length, doubleword write bursts on PLB to SDRAM. The final write burst is fixed-length, since the PCI write has completed, and PCI bridge knows the exact burst length.

21.10.1.3 CPU Read From PCI Memory Slave, Nonprefetching

In *Figure 21-69 CPU Read From PCI Memory Slave, Nonprefetching* on page 511 and *Figure 21-96 CPU Read From PCI Memory Slave, Nonprefetching* on page 538, a PLB Master (CPU) executes a single-beat 64-bit read from a region of PCI memory marked as nonprefetchable. PCI bridge responds as a 32-bit PLB slave, so the CPU executes conversion cycles for each read. PCI bridge executes a single-beat PCI read for both PLB read requests.

21.10.1.4 CPU Read From PCI Memory Slave, Prefetching

In *Figure 21-71 CPU Read From PCI Memory Slave, Prefetching* on page 513 and *Figure 21-98 CPU Read From PCI Memory Slave, Prefetching* on page 540, a PLB Master (CPU) executes 8, 64-bit, single-beat reads from a region of PCI memory marked as prefetchable. The first PLB read causes PCI bridge to execute a 64-byte Read Multiple to fill its 64-byte read prefetch buffer. The data for subsequent PLB reads is provided from the read buffer and no PCI cycles are generated. PCI bridge responds as a 32-bit PLB slave, so the CPU executes conversion cycles for each read.

21.10.1.5 CPU Write To PCI Memory Slave

Figure 21-75 CPU Write To PCI Memory Slave on page 517 and *Figure 21-102 CPU Write To PCI Memory Slave* on page 544 show a PLB Master (CPU) executing 4, 64-bit, single-beat writes to PCI memory. PCI bridge responds as a 32-bit slave, so the CPU executes conversion cycles for each write. PCI bridge posts the writes in its 4-entry write buffer, and executes a PCI single-beat Memory Write for each request.

21.10.1.6 PCI Memory To SDRAM DMA Transfer

Figure 21-78 PCI Memory To SDRAM DMA Transfer on page 520 and *Figure 21-107 PCI Memory To SDRAM DMA Transfer (Continued)* on page 549 show a DMA transfer of data from PCI memory to SDRAM. The DMA PLB Master executes a 4-doubleword read burst to PCI bridge followed by a 4-doubleword write burst to SDRAM. For the read, PCI bridge executes a 32-byte PCI Read Line.

21.10.1.7 SDRAM To PCI Memory DMA Transfer

Figure 21-80 SDRAM To PCI Memory DMA Transfer on page 522 and *Figure 21-109 SDRAM To PCI Memory DMA Transfer* on page 551 show a DMA transfer of data from SDRAM to PCI memory. The DMA PLB Master executes a 4-doubleword read burst from SDRAM followed by a 4-doubleword write burst to PCI memory. PCI bridge then executes a 32-byte write on the PCI bus.

21.10.2 Asynchronous

The following diagrams are for asynchronous clocking mode. Note that all of the diagrams flow across multiple pages. Each diagram begins with cycle 1 on the left facing page.

Figure 21-61. PCI Master Burst Read From SDRAM

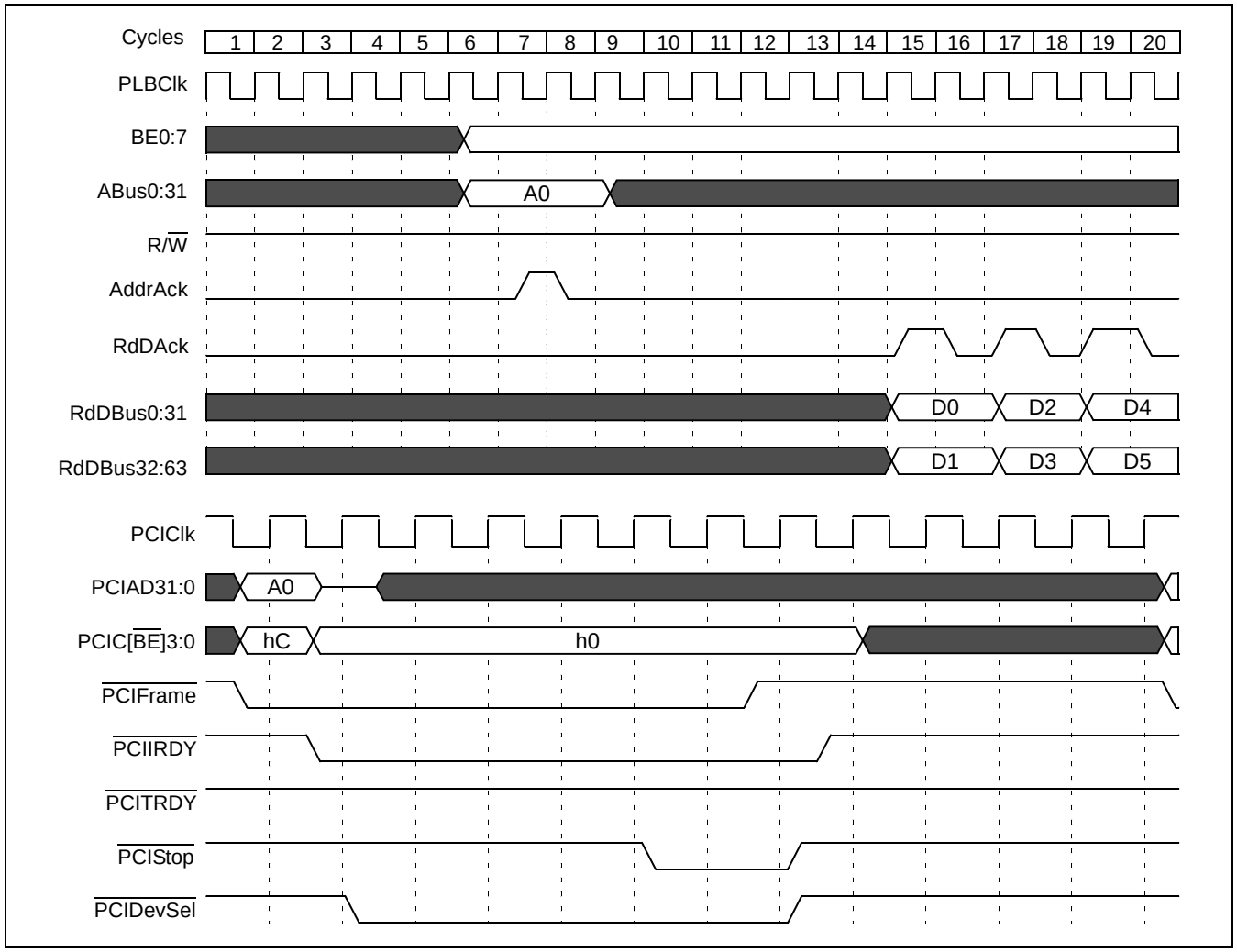


Figure 21-62. PCI Master Burst Read From SDRAM (Continued)

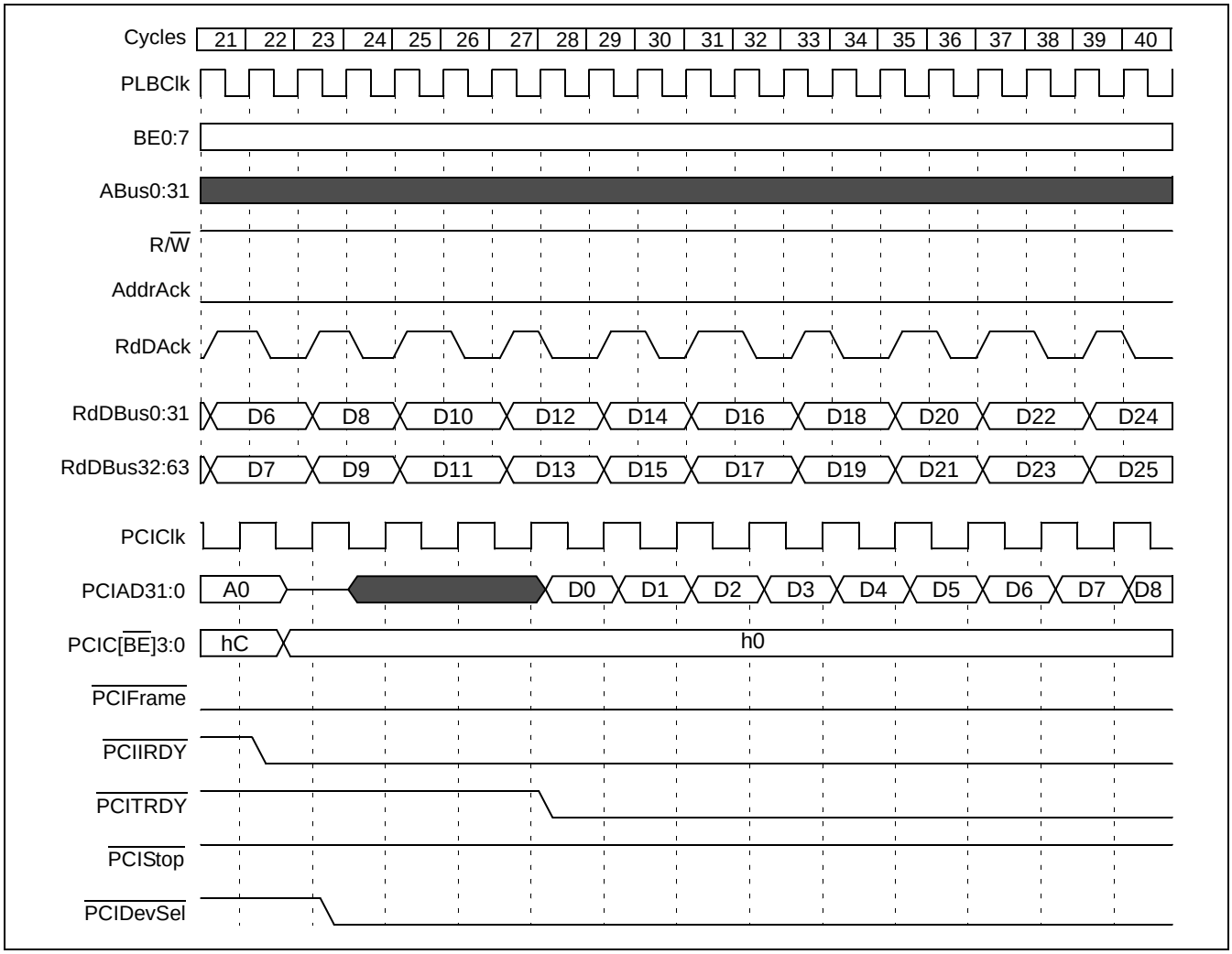


Figure 21-63. PCI Master Burst Read From SDRAM (Continued)

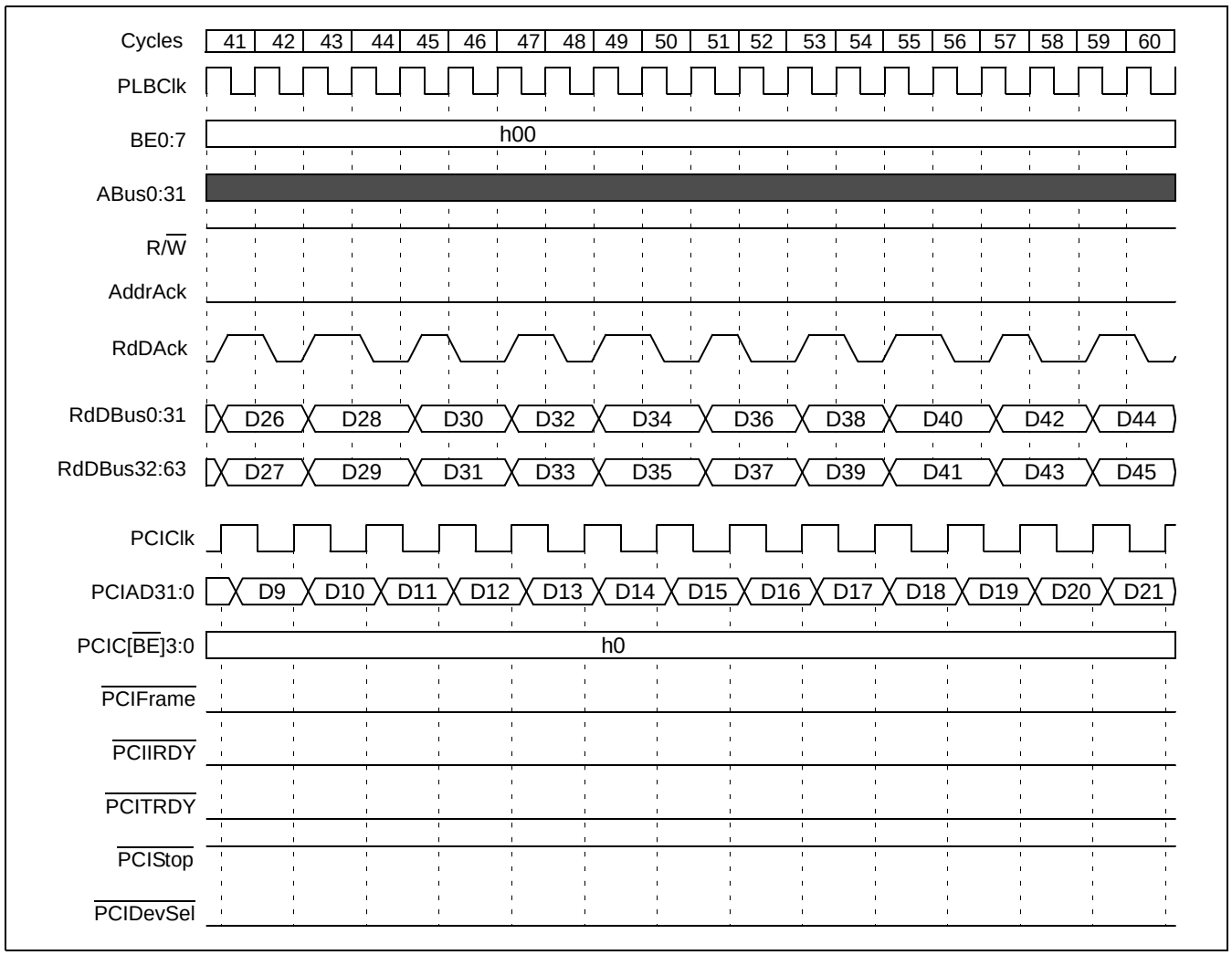


Figure 21-64. PCI Master Burst Read From SDRAM (Continued)

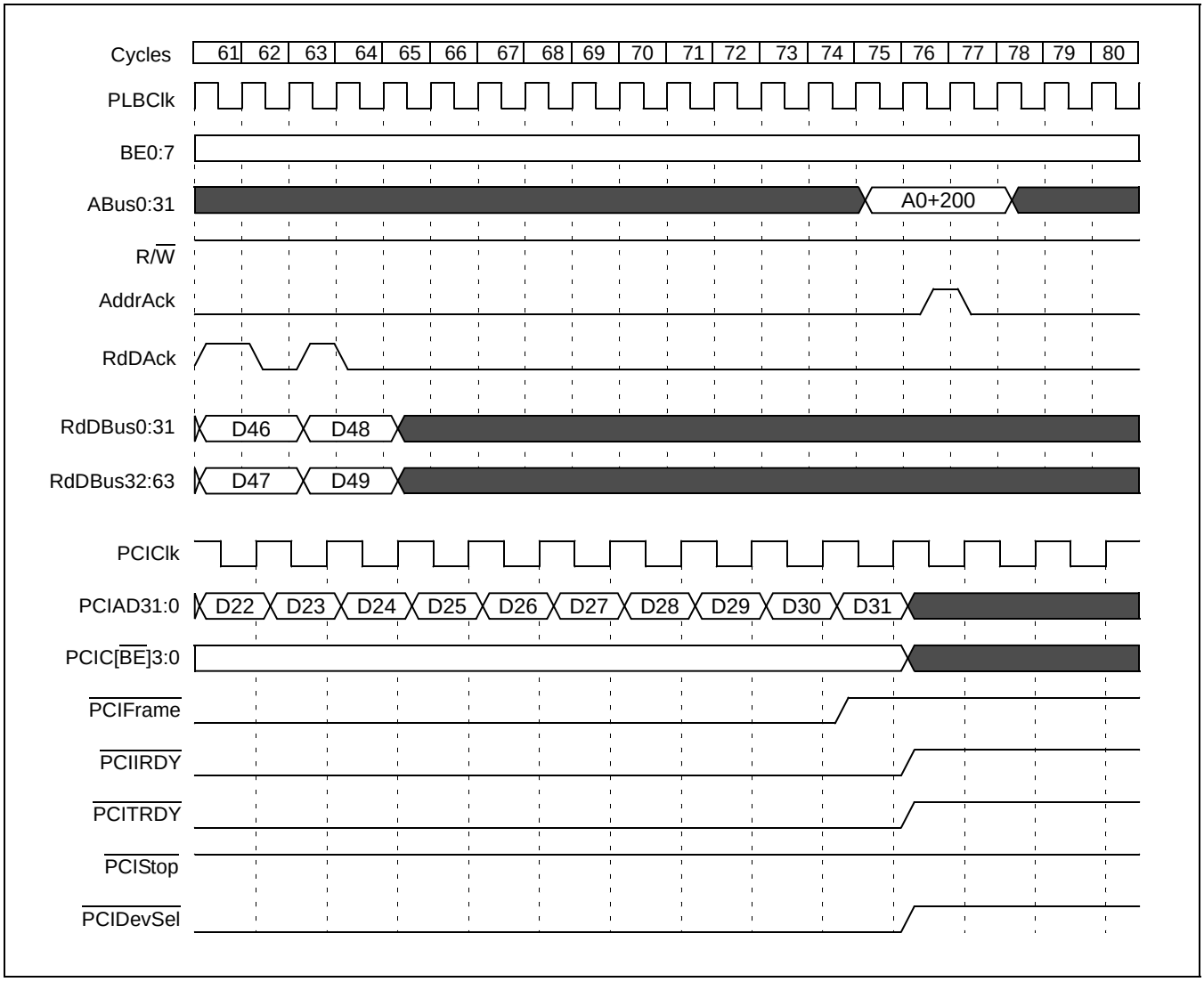


Figure 21-65. PCI Master Burst Write To SDRAM

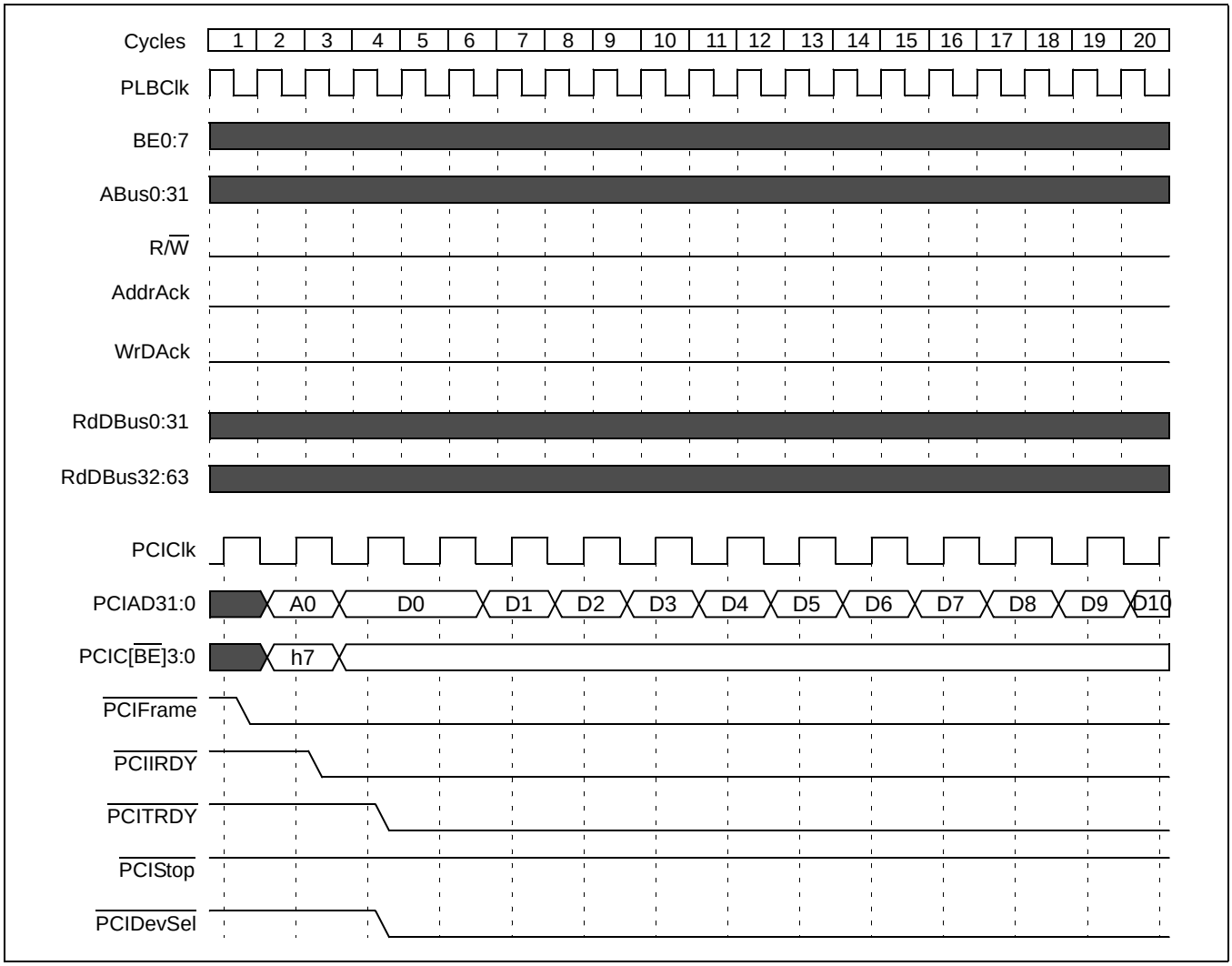


Figure 21-66. PCI Master Burst Write To SDRAM (Continued)

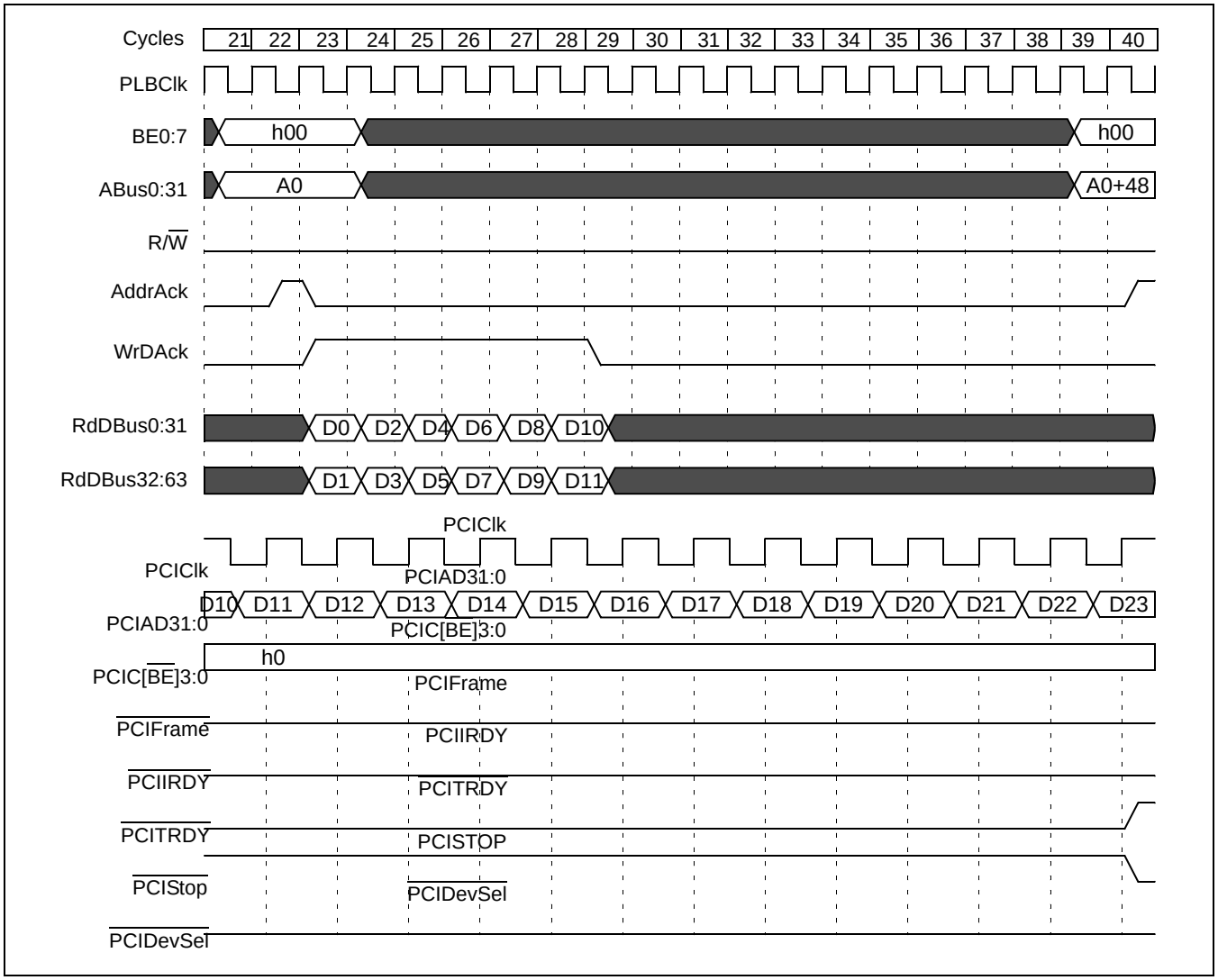


Figure 21-67. PCI Master Burst Write To SDRAM (Continued)

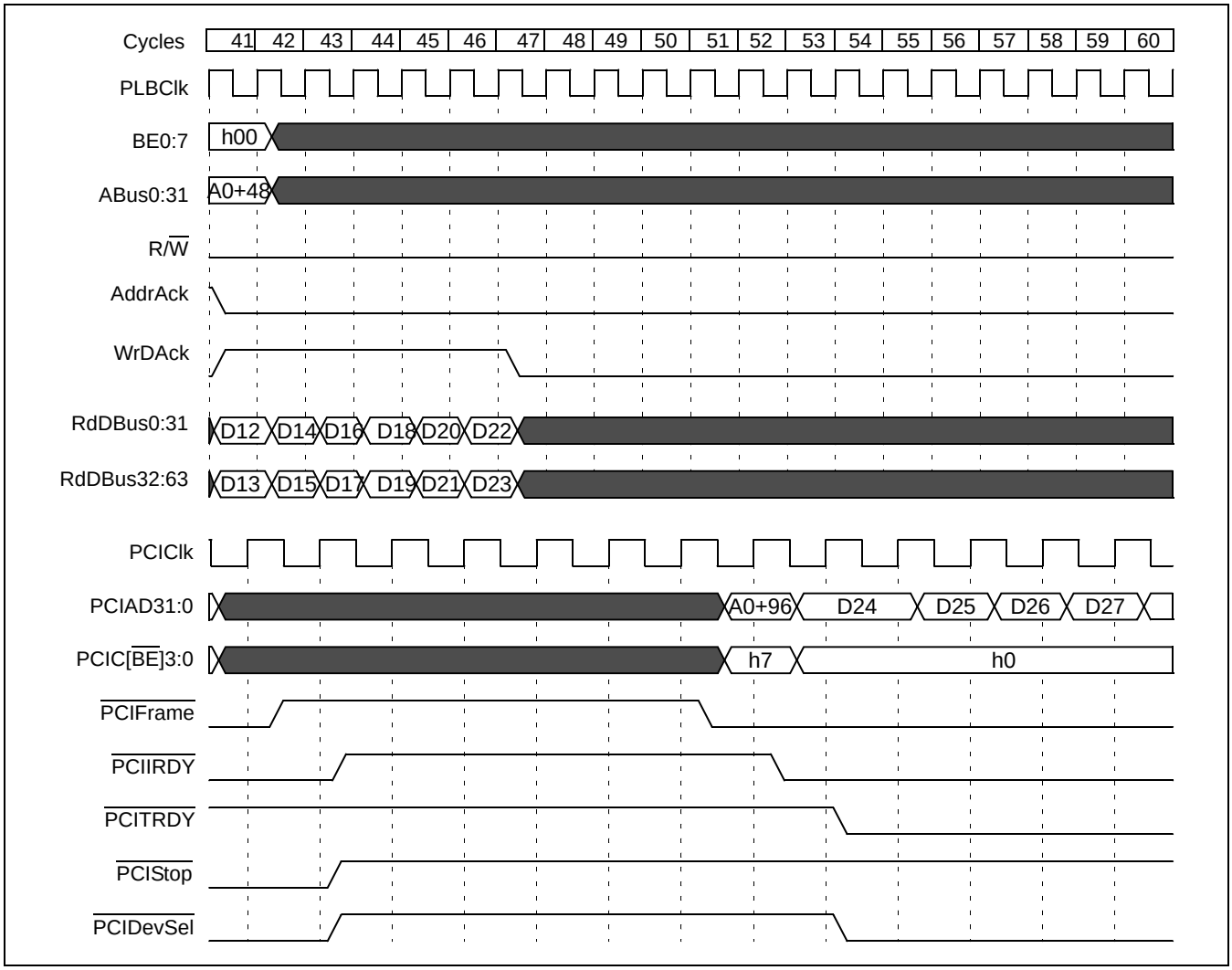


Figure 21-68. PCI Master Burst Write To SDRAM (Continued)

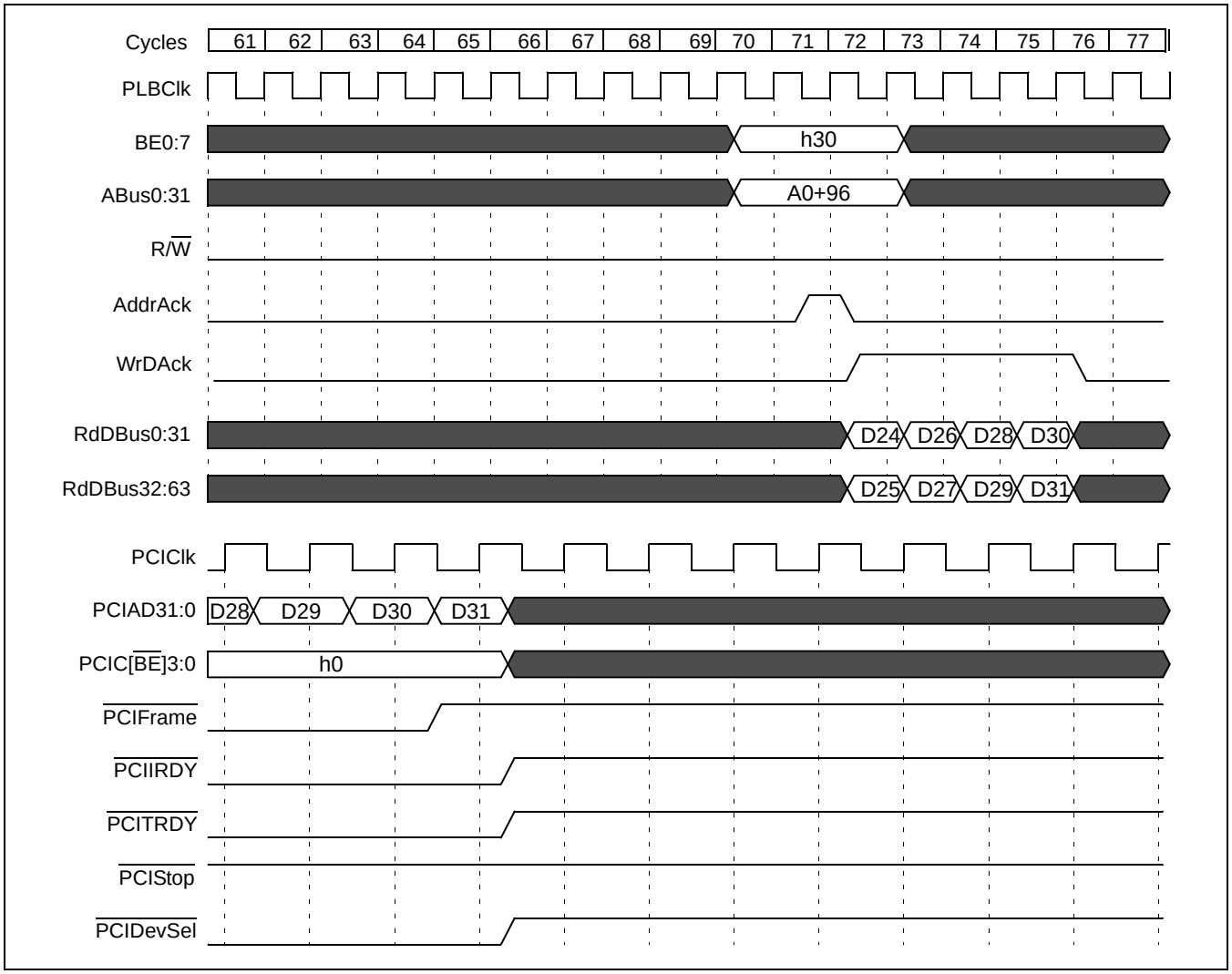


Figure 21-69. CPU Read From PCI Memory Slave, Nonprefetching

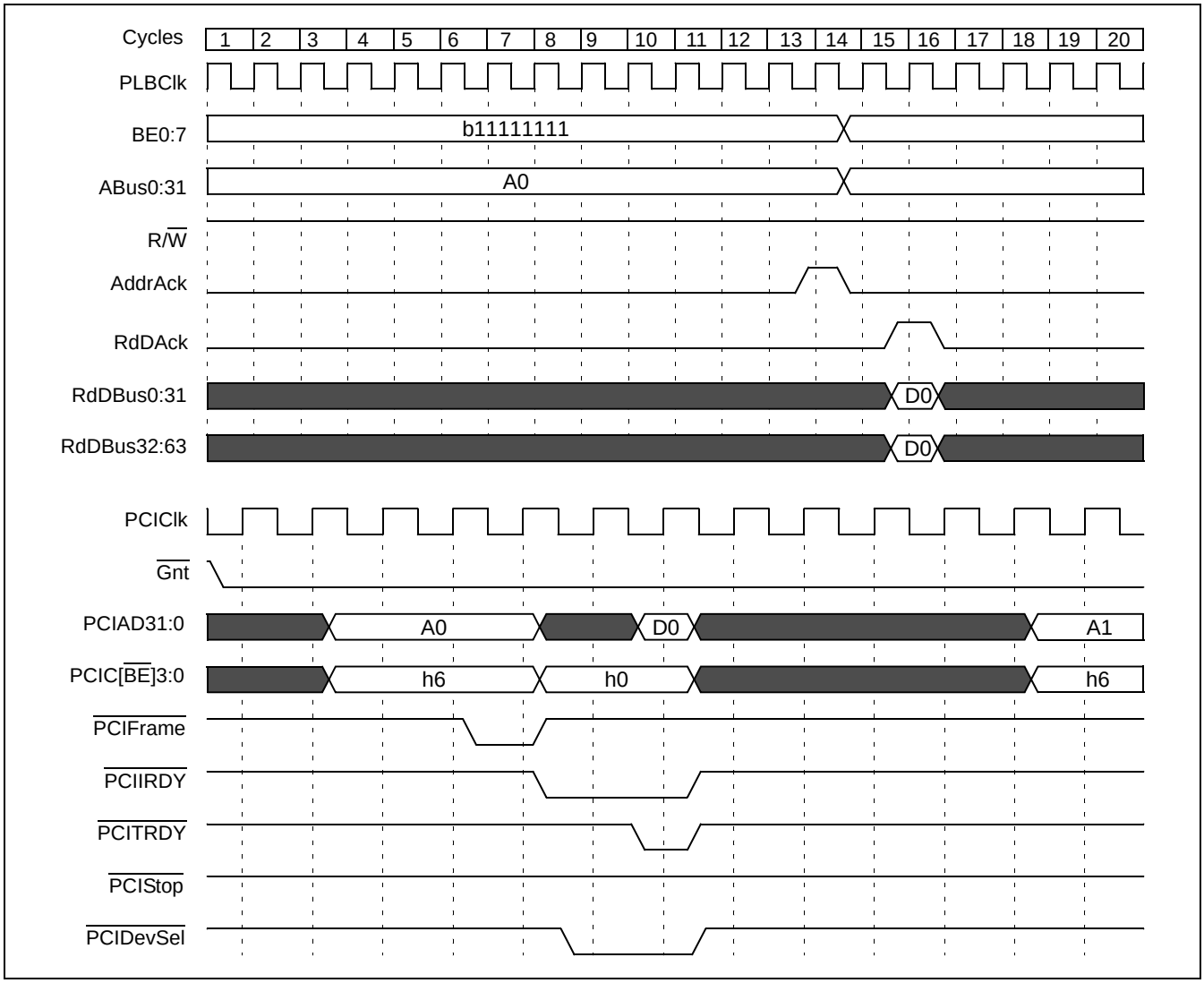


Figure 21-70. CPU Read From PCI Memory Slave, Nonprefetching (Continued)

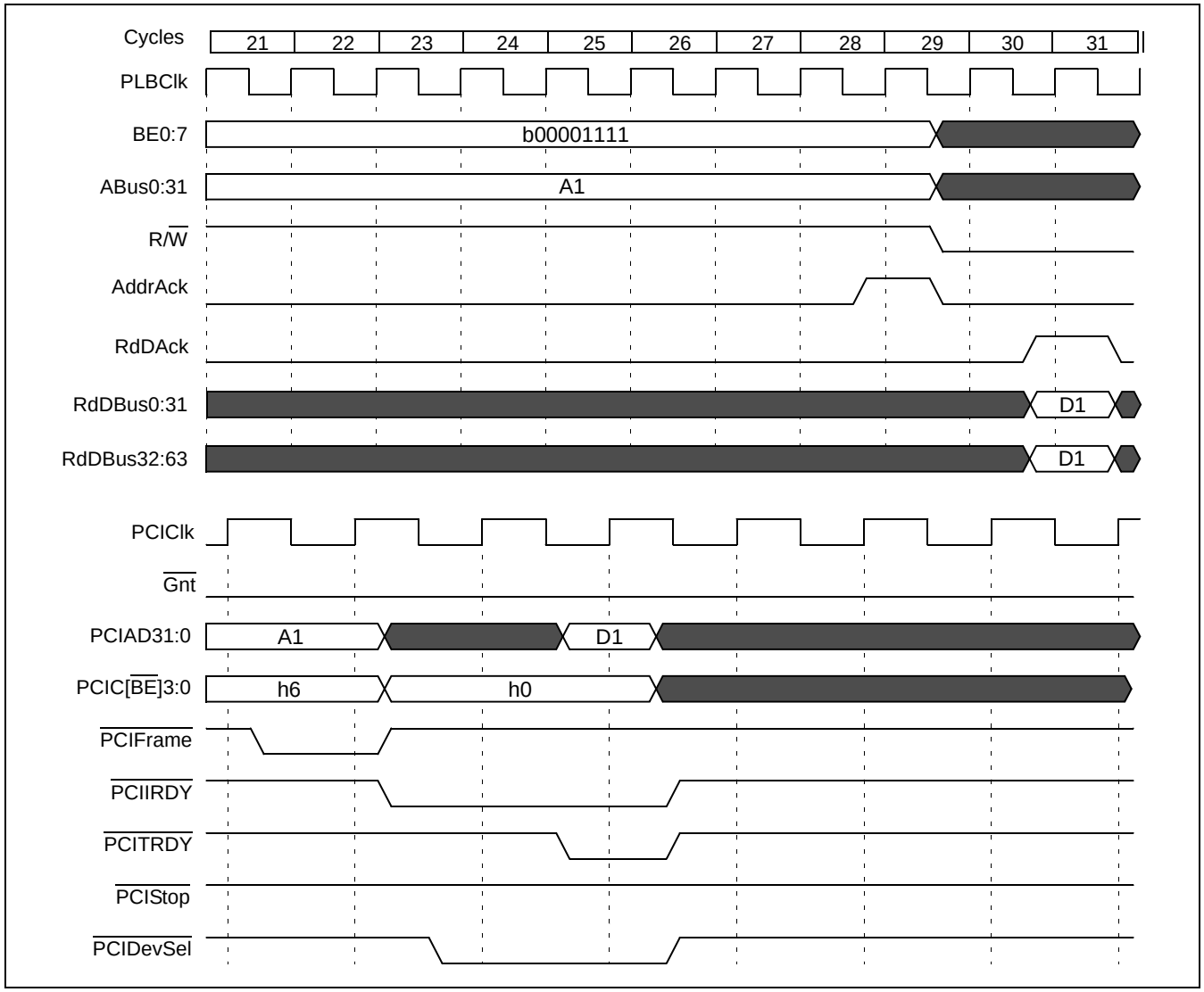


Figure 21-71. CPU Read From PCI Memory Slave, Prefetching

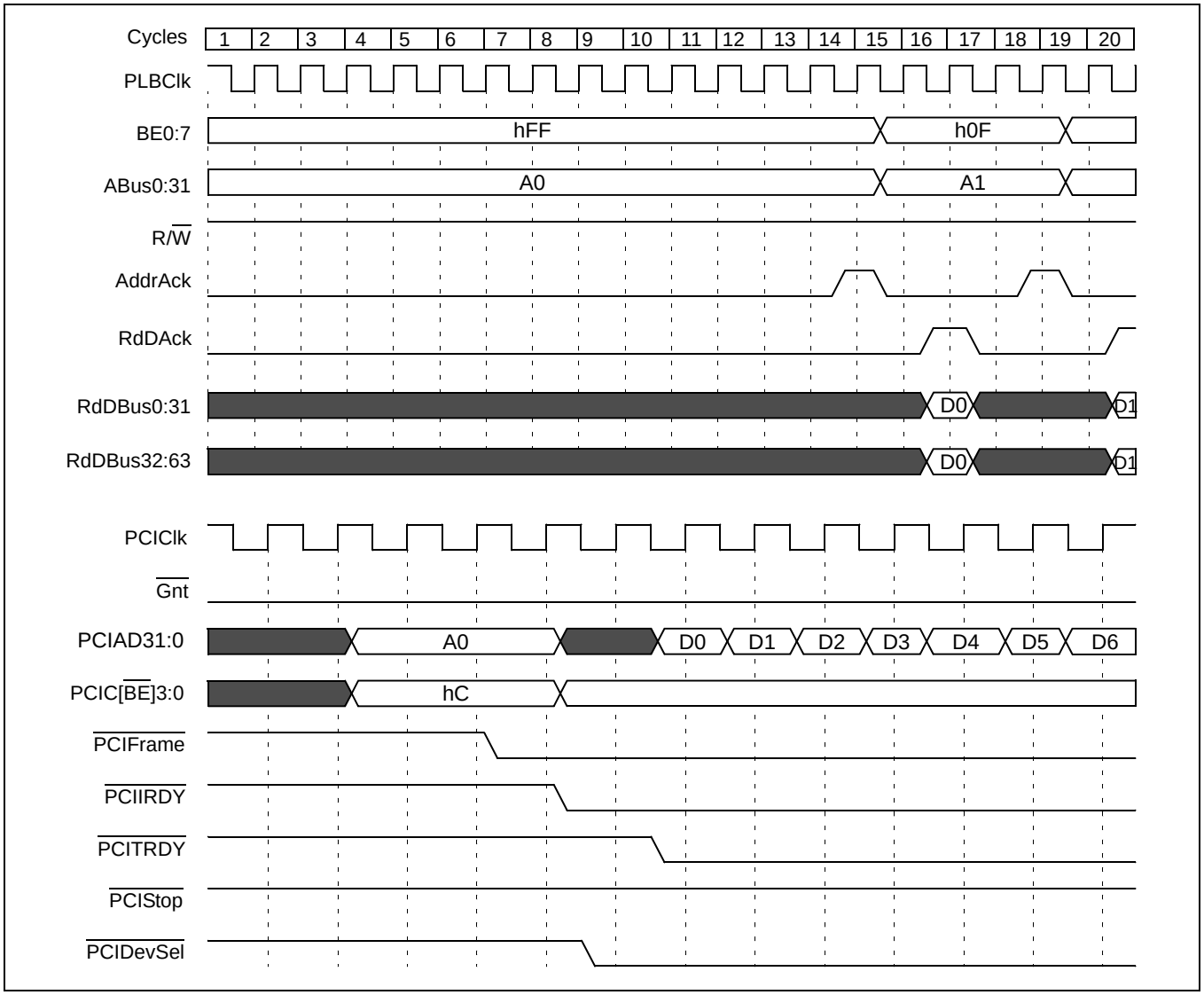


Figure 21-72. CPU Read From PCI Memory Slave, Prefetching (Continued)

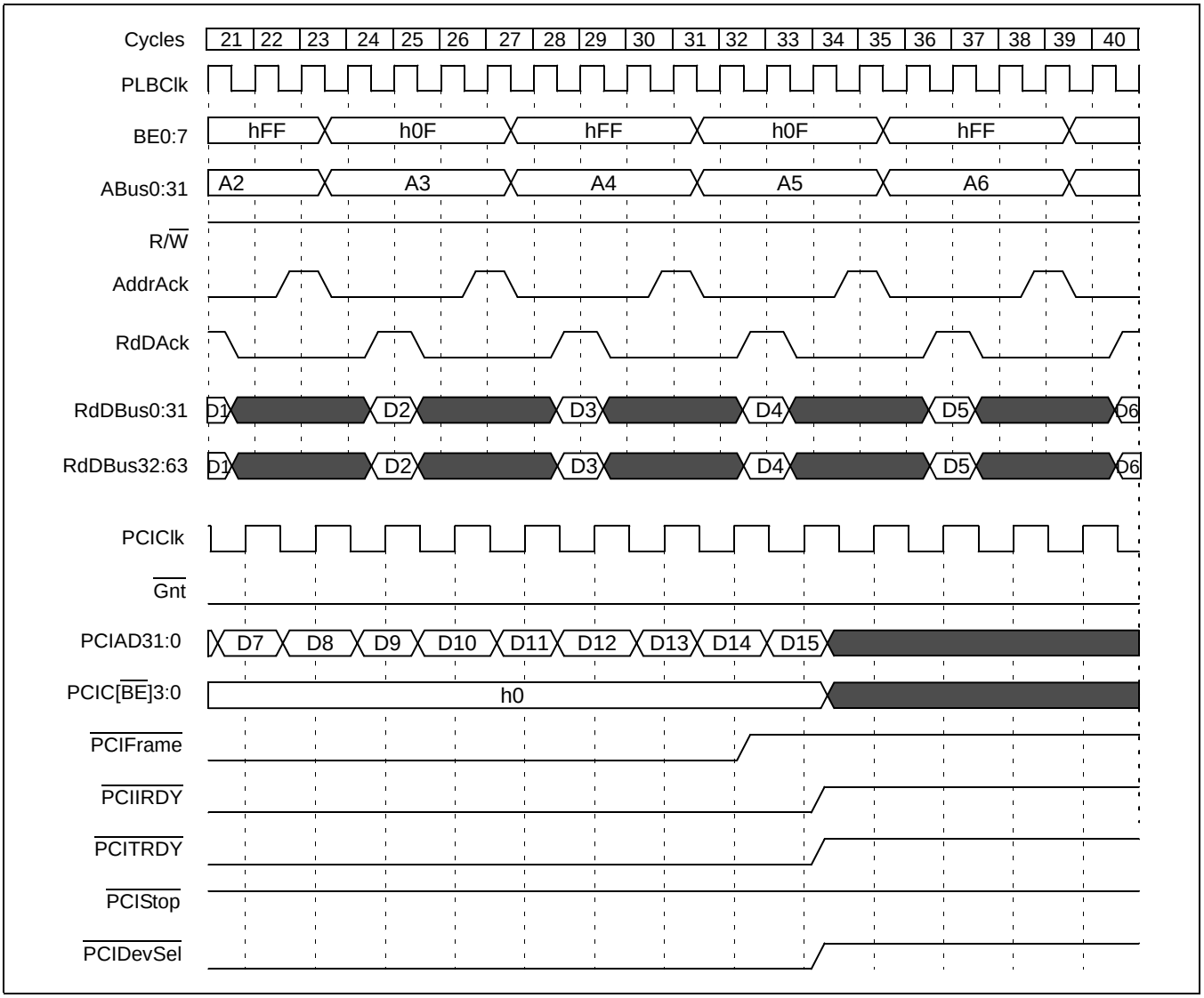


Figure 21-73. CPU Read From PCI Memory Slave, Prefetching (Continued)

Figure 21-74. CPU Read From PCI Memory Slave, Prefetching (Continued)

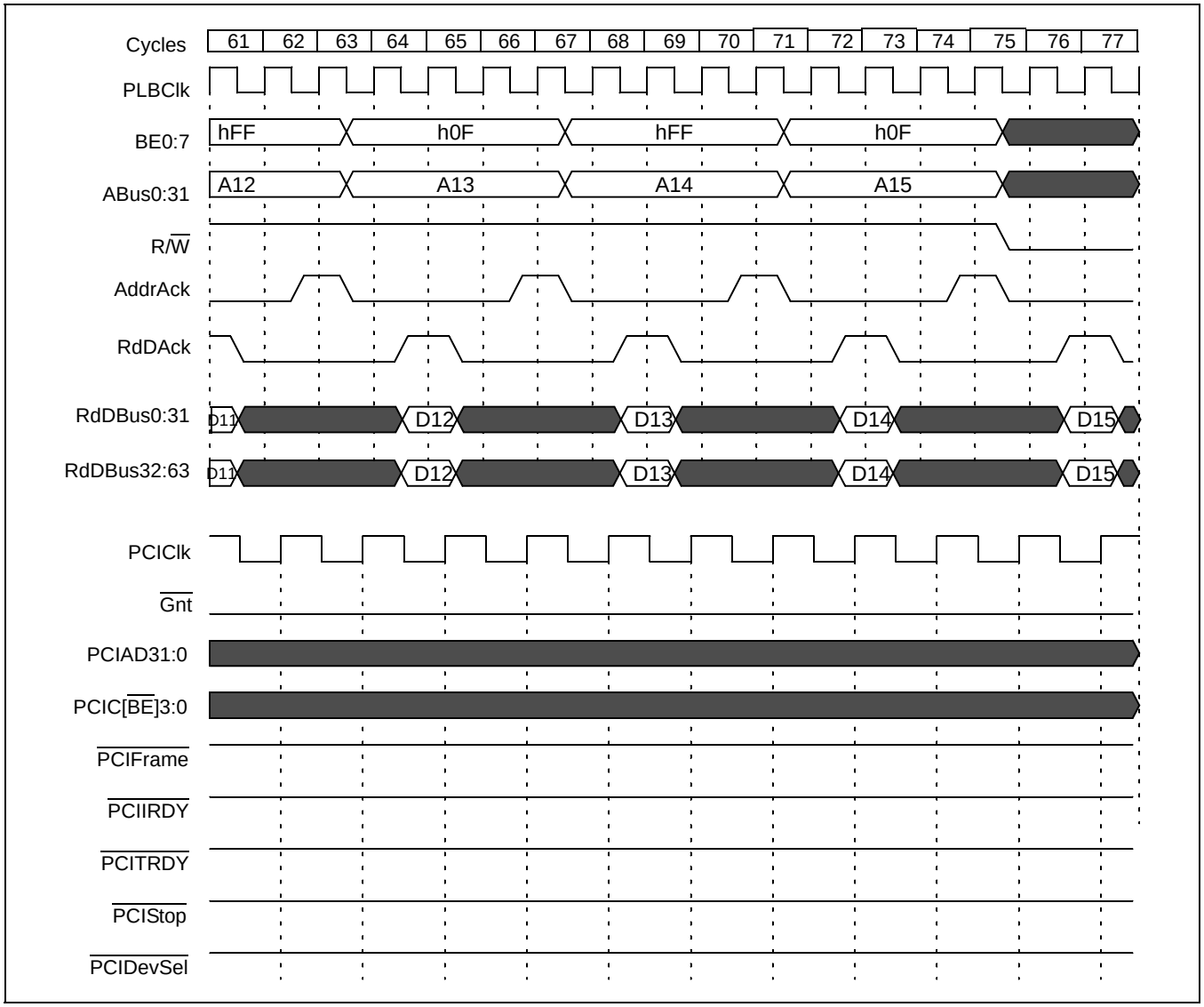


Figure 21-75. CPU Write To PCI Memory Slave

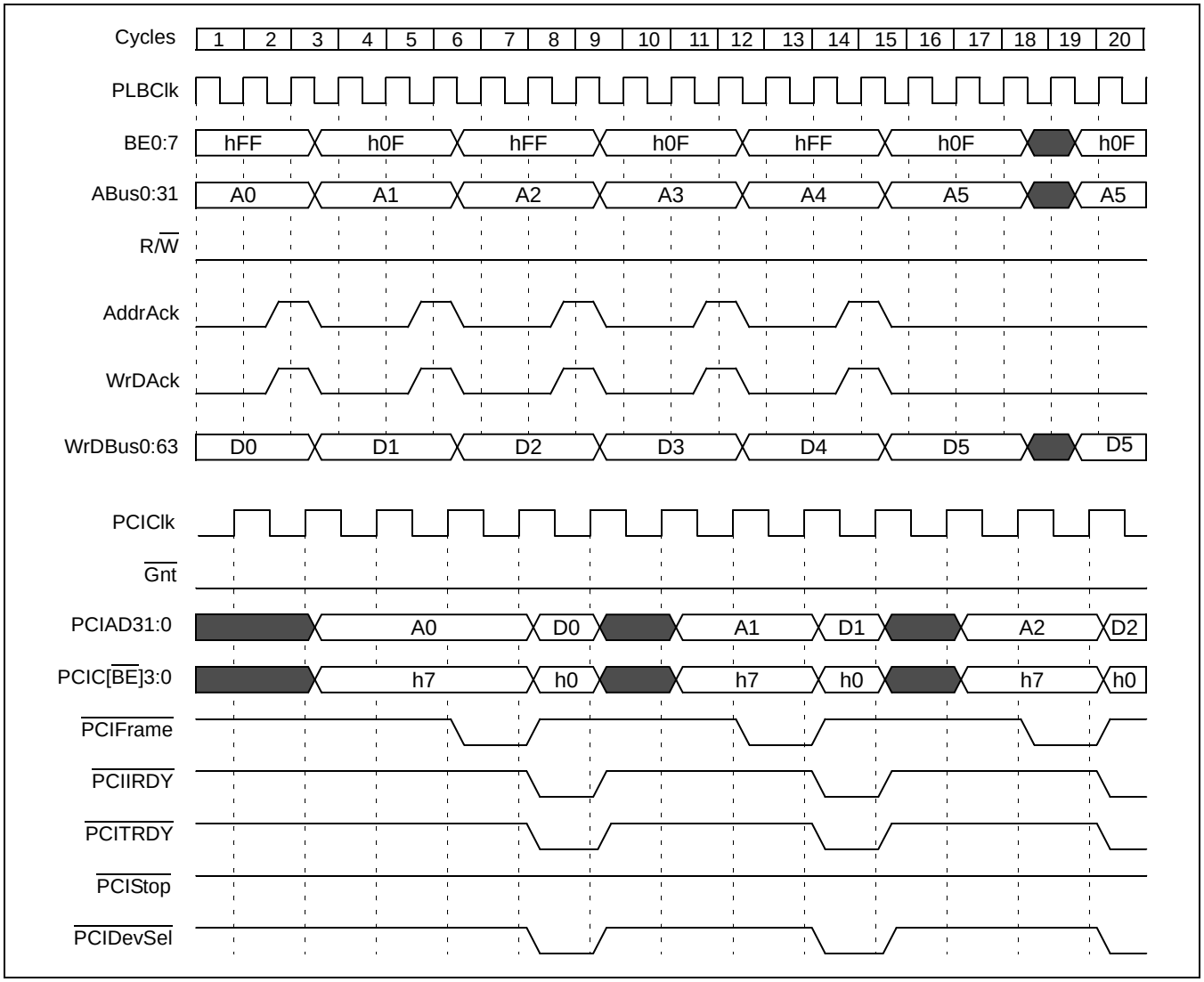


Figure 21-76. CPU Write To PCI Memory Slave (Continued)

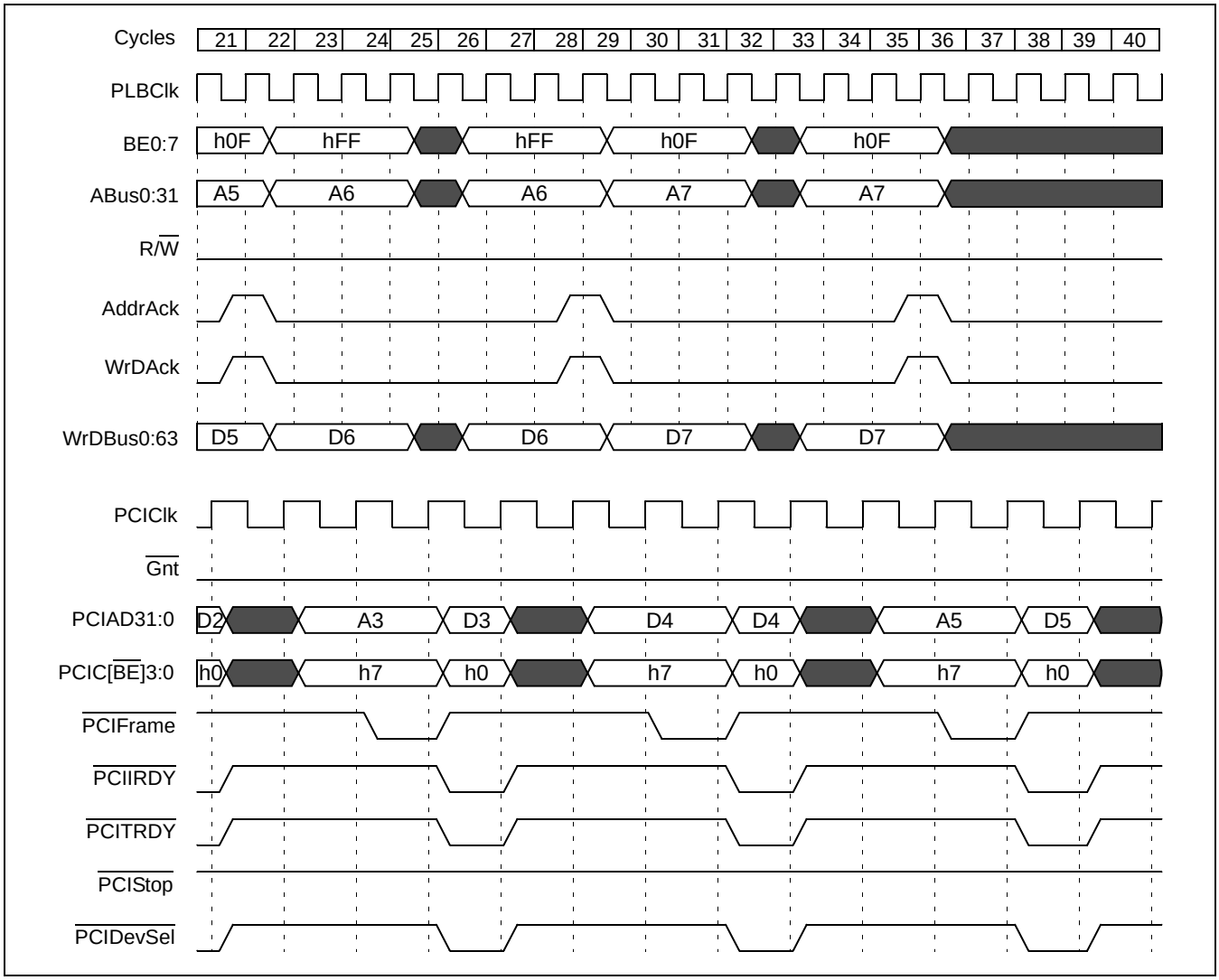


Figure 21-77. CPU Write To PCI Memory Slave (Continued)

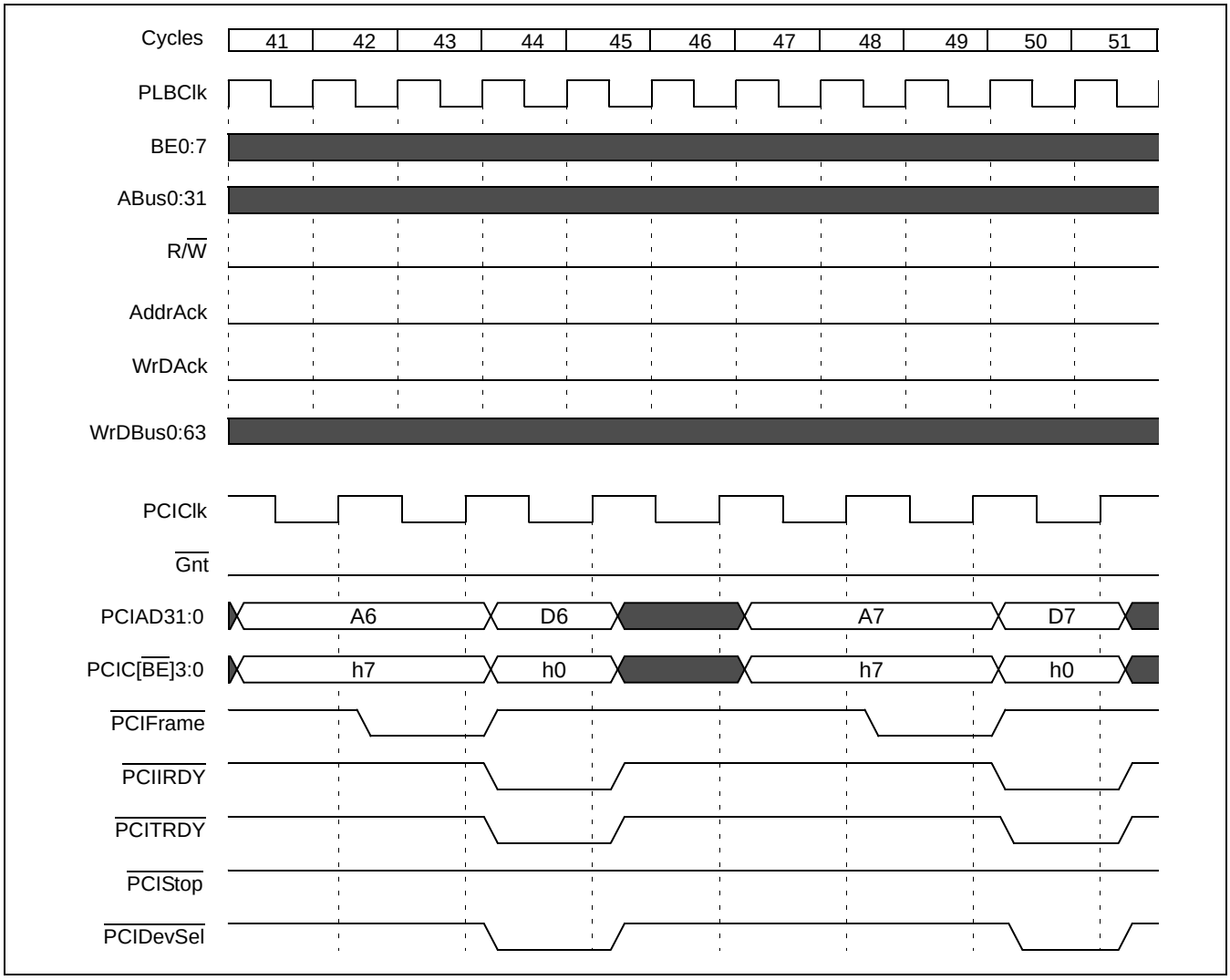


Figure 21-78. PCI Memory To SDRAM DMA Transfer

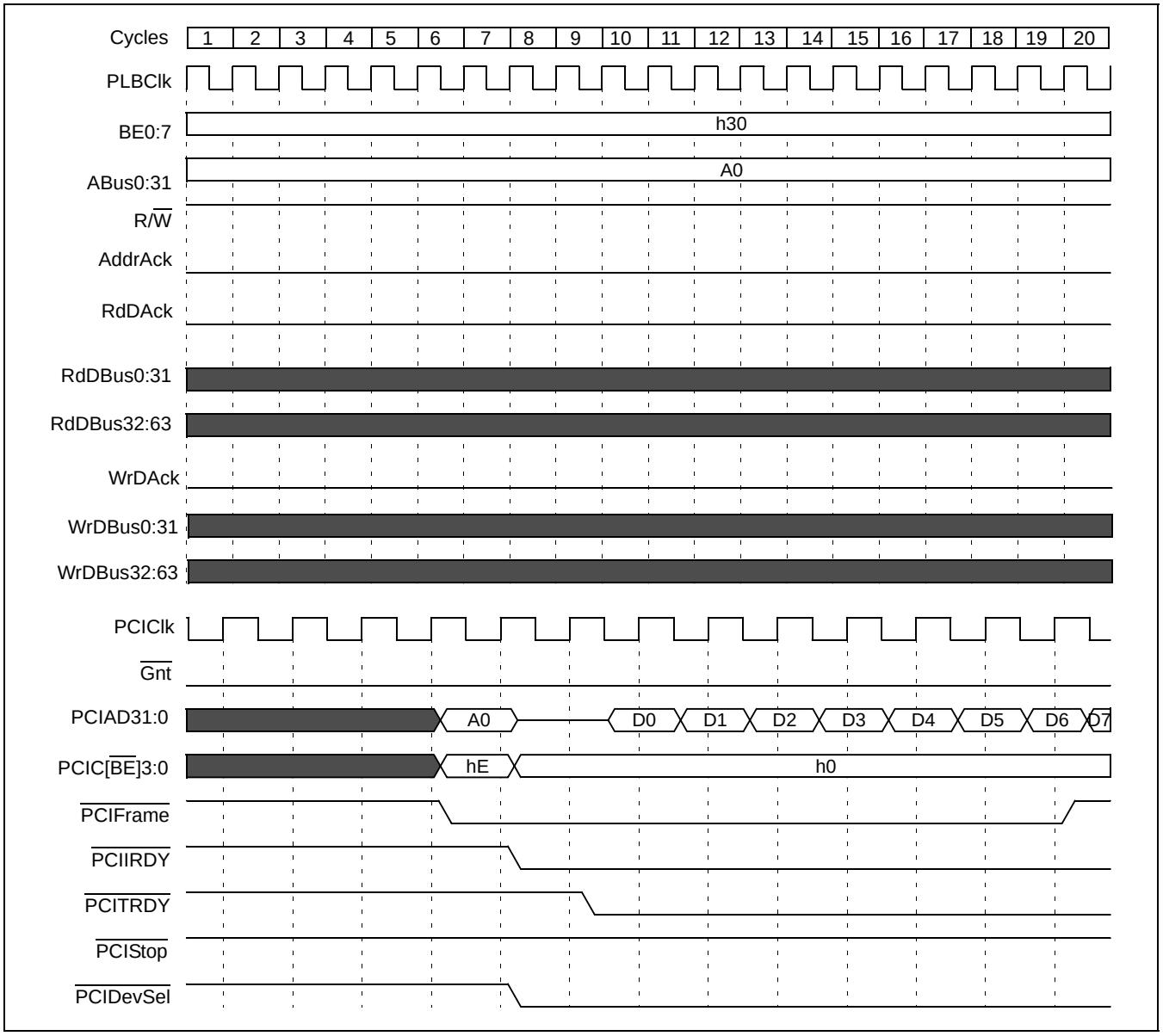


Figure 21-79. PCI Memory To SDRAM DMA Transfer (Continued)

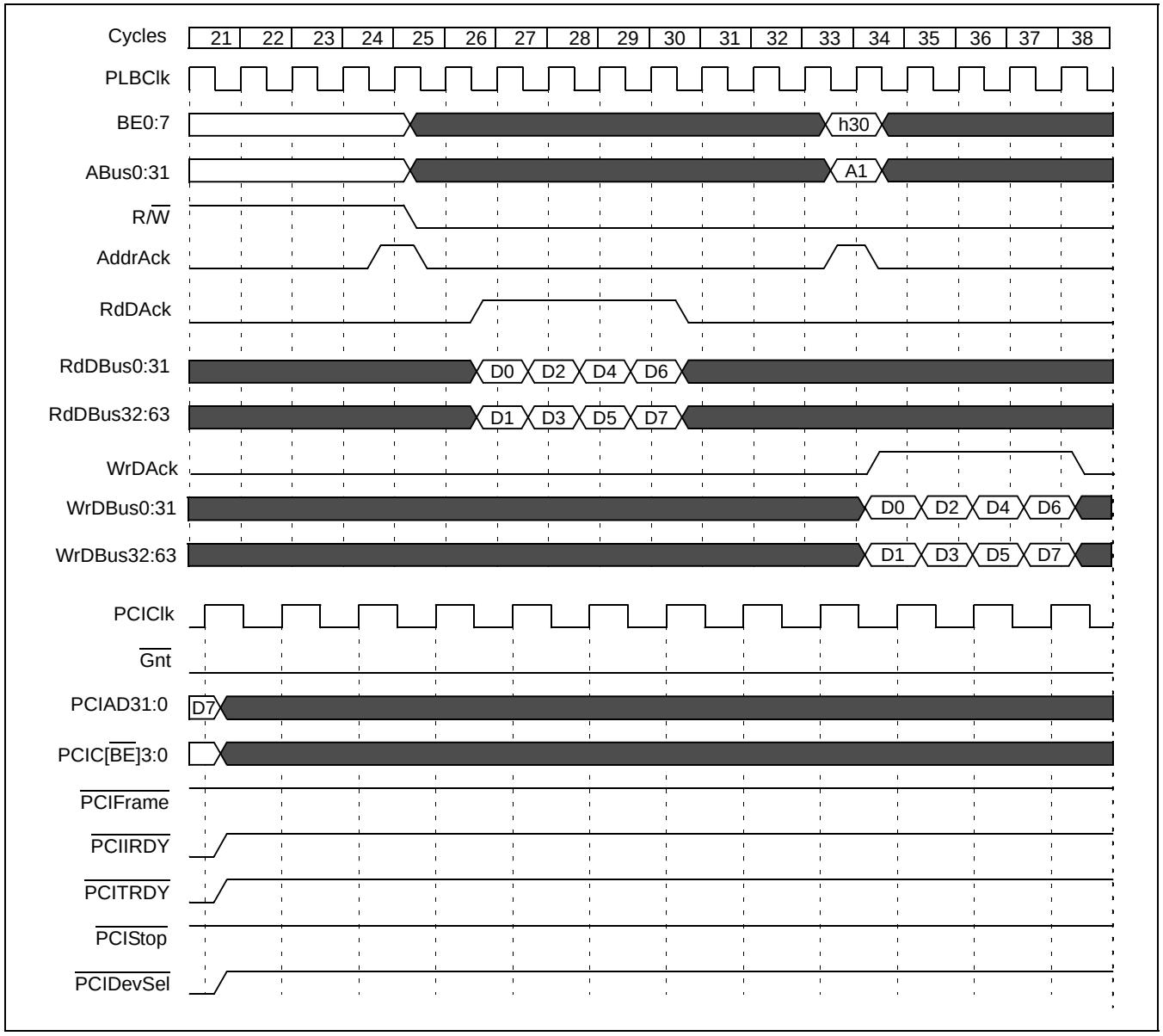


Figure 21-80. SDRAM To PCI Memory DMA Transfer

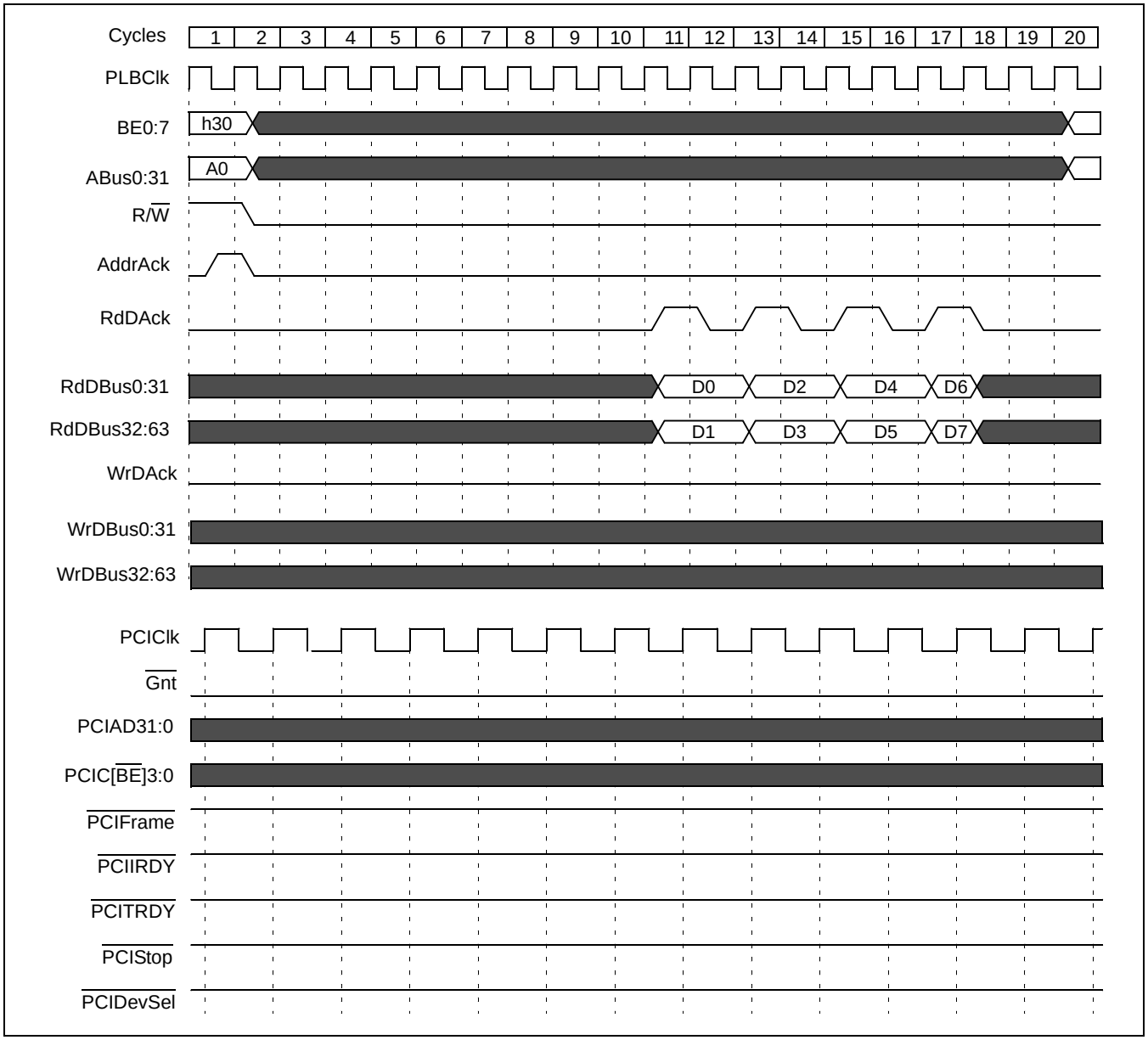


Figure 21-81. SDRAM To PCI Memory DMA Transfer (Continued)

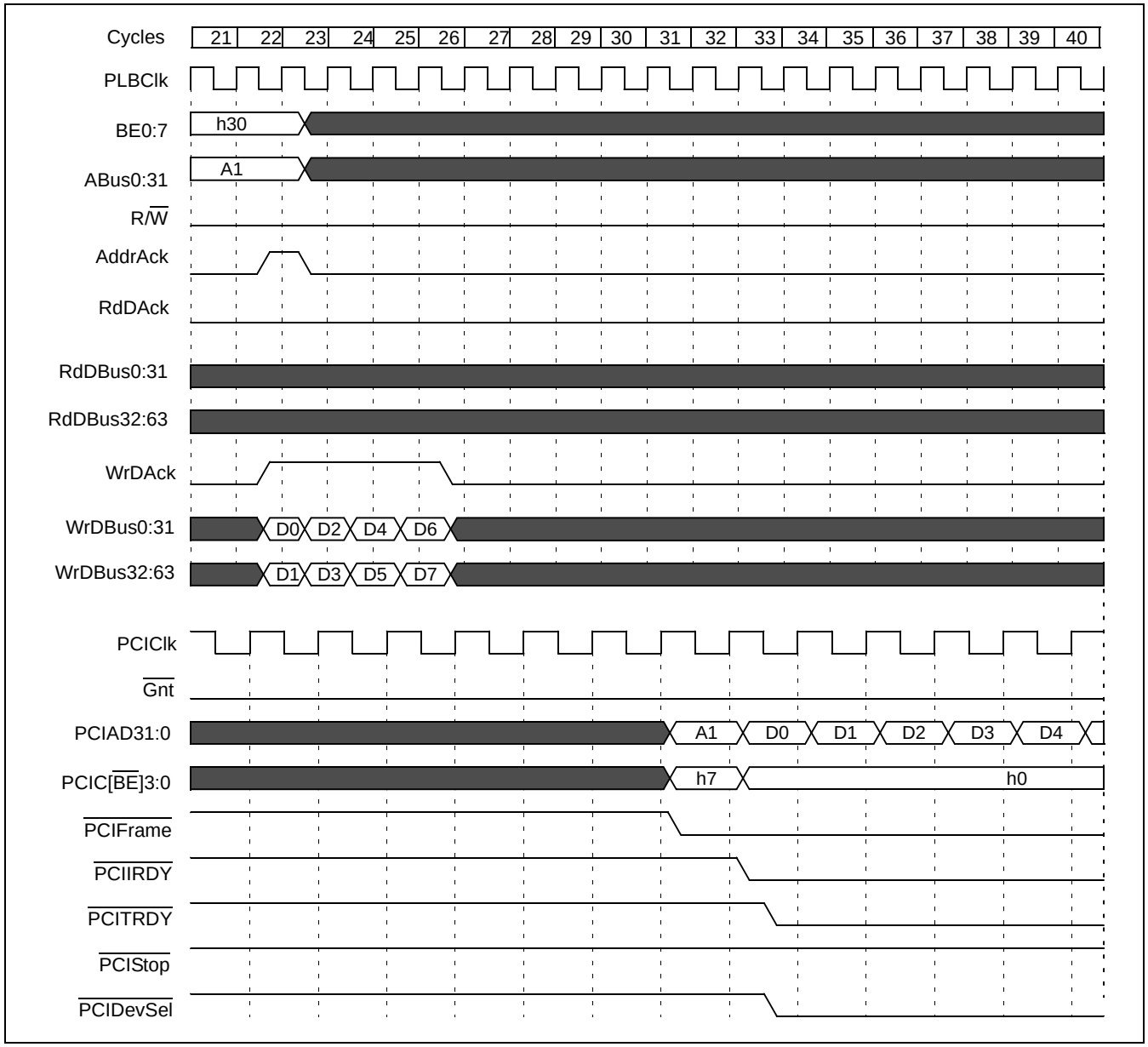
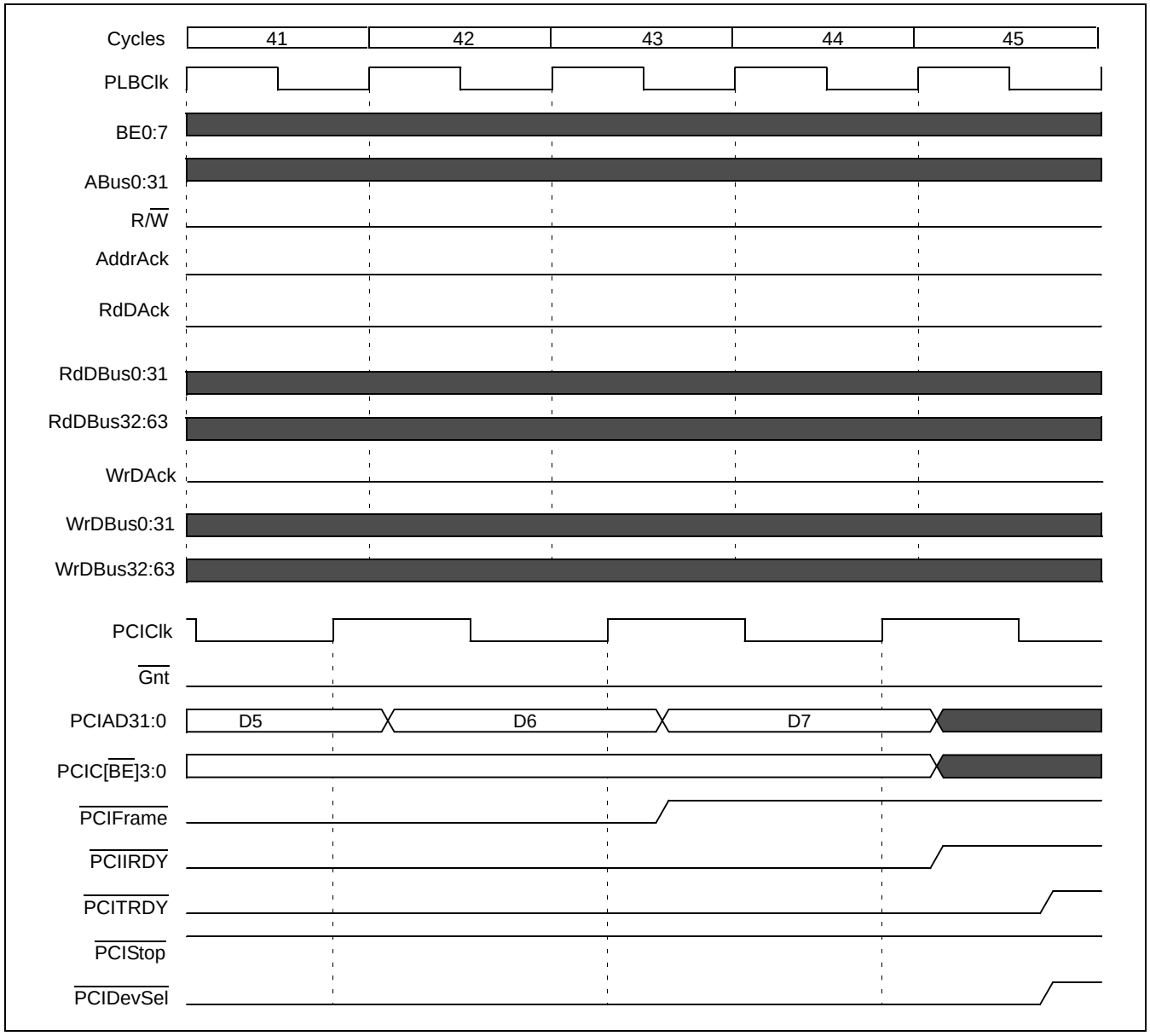


Figure 21-82. SDRAM To PCI Memory DMA Transfer (Continued)



21.10.3 Synchronous

The following diagrams are for synchronous clocking mode. Note that all of the diagrams flow across multiple pages. Each diagram begins with cycle 1 on the left facing page.

Figure 21-83. PCI Master Burst Read From SDRAM

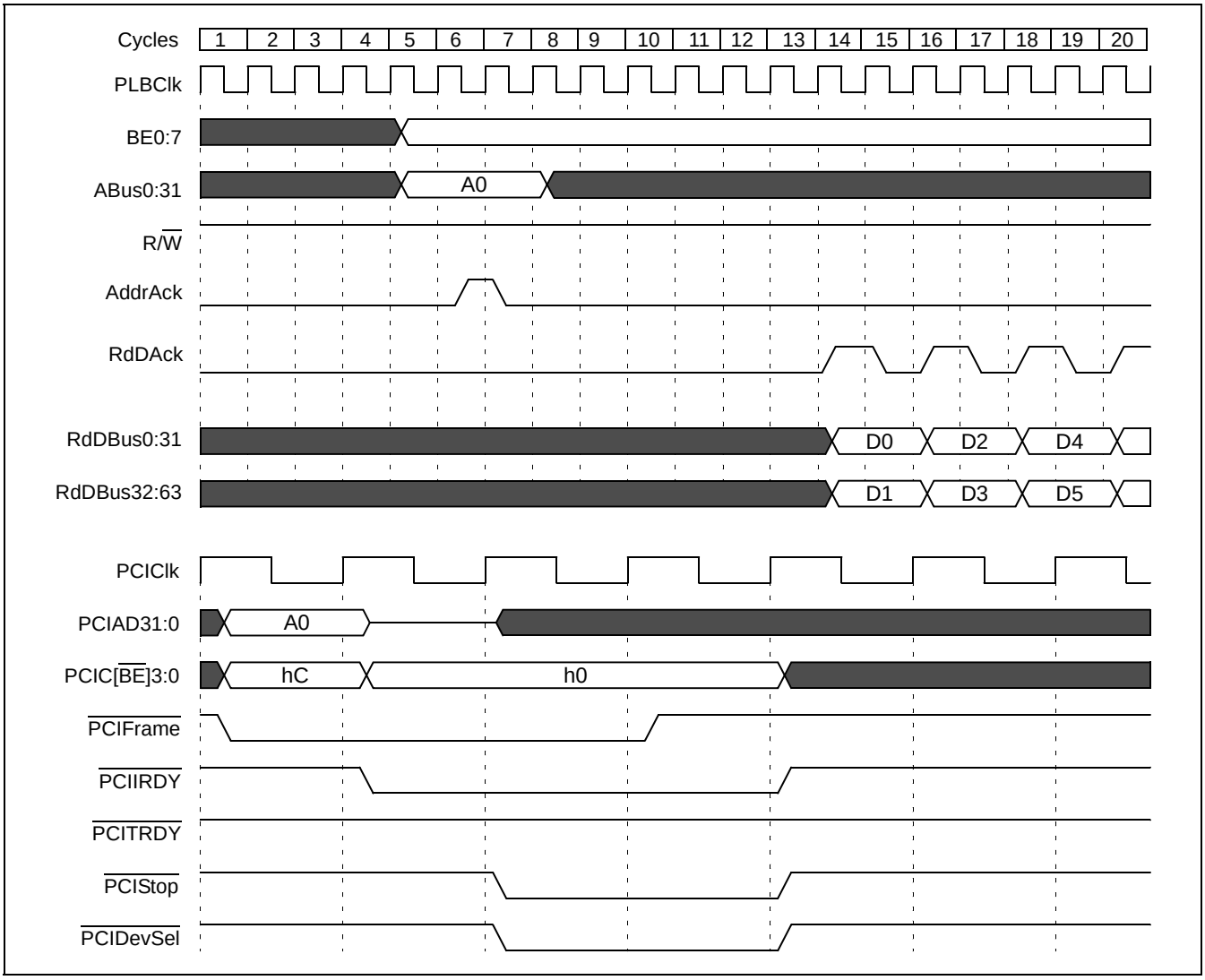


Figure 21-84. PCI Master Burst Read From SDRAM (Continued)

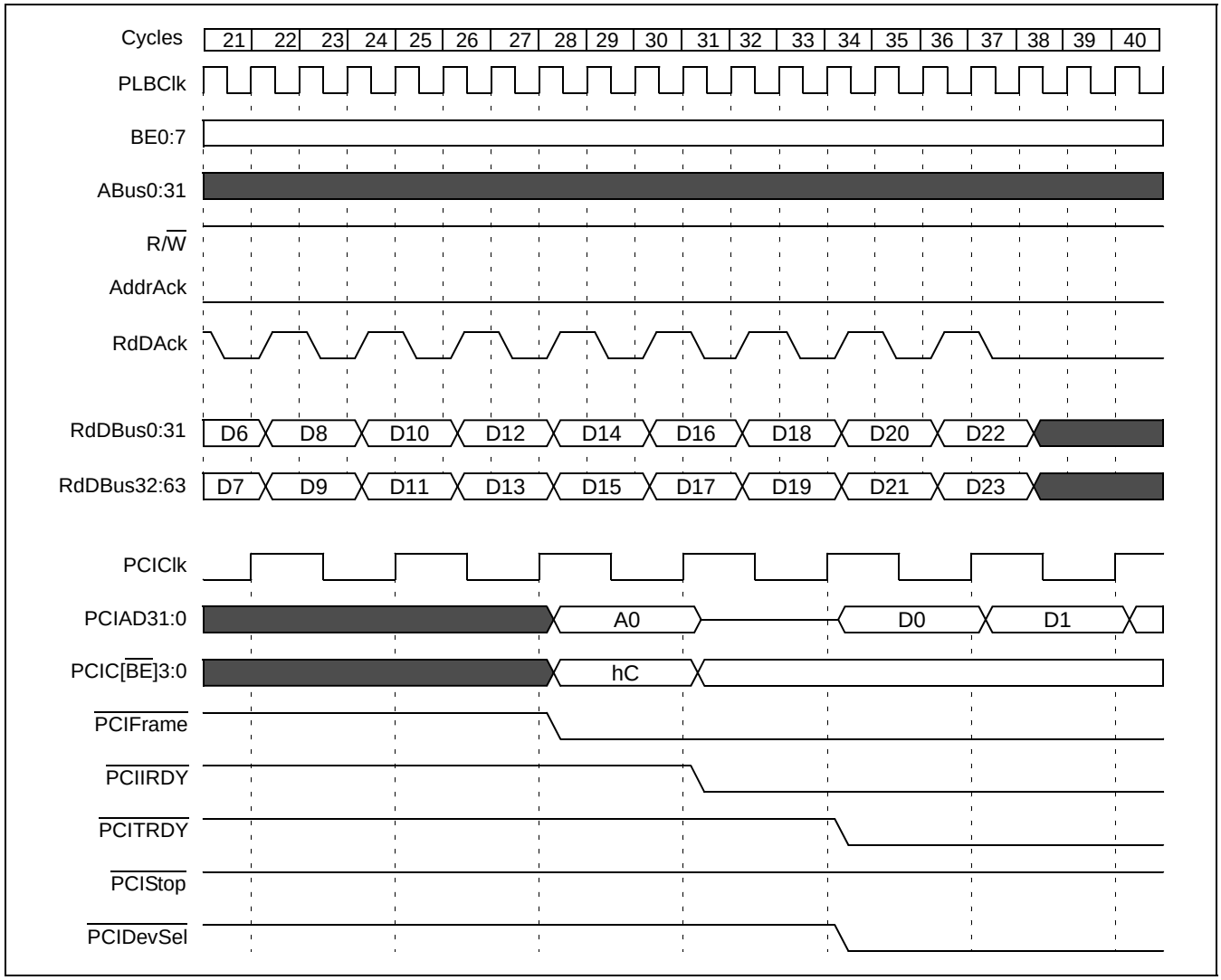


Figure 21-85. PCI Master Burst Read From SDRAM (Continued)

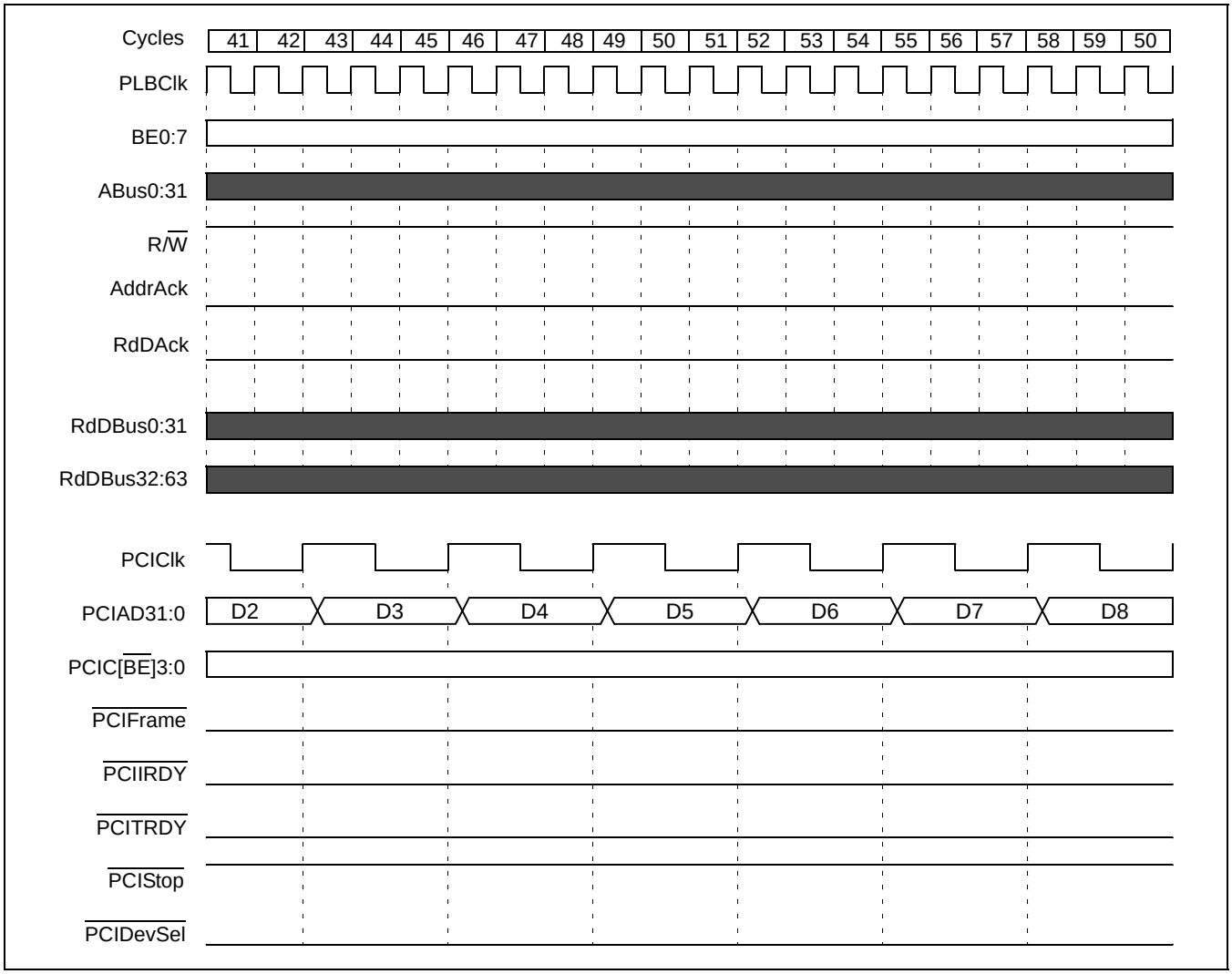


Figure 21-86. PCI Master Burst Read From SDRAM (Continued)

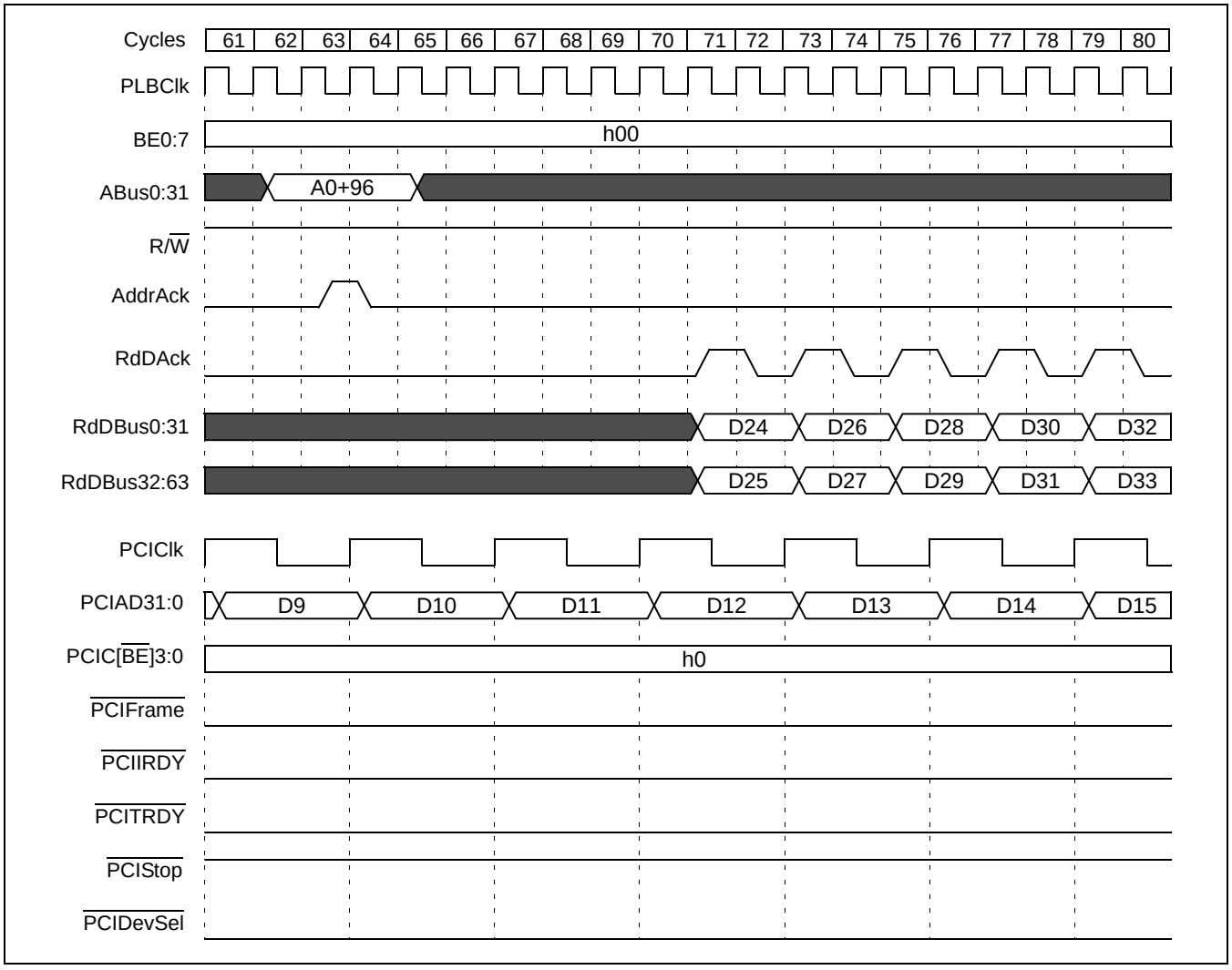


Figure 21-87. PCI Master Burst Read From SDRAM (Continued)

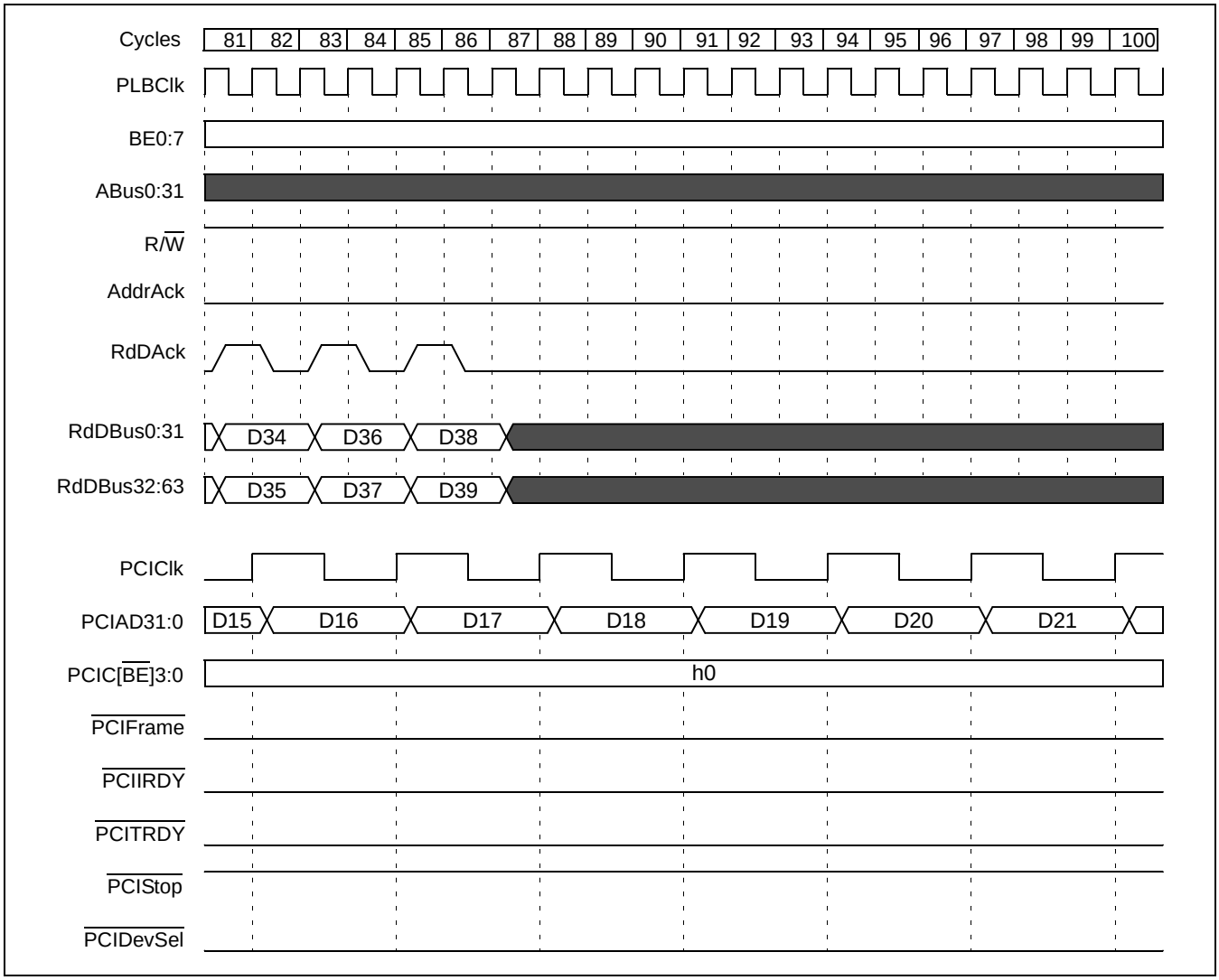


Figure 21-88. PCI Master Burst Read From SDRAM (Continued)

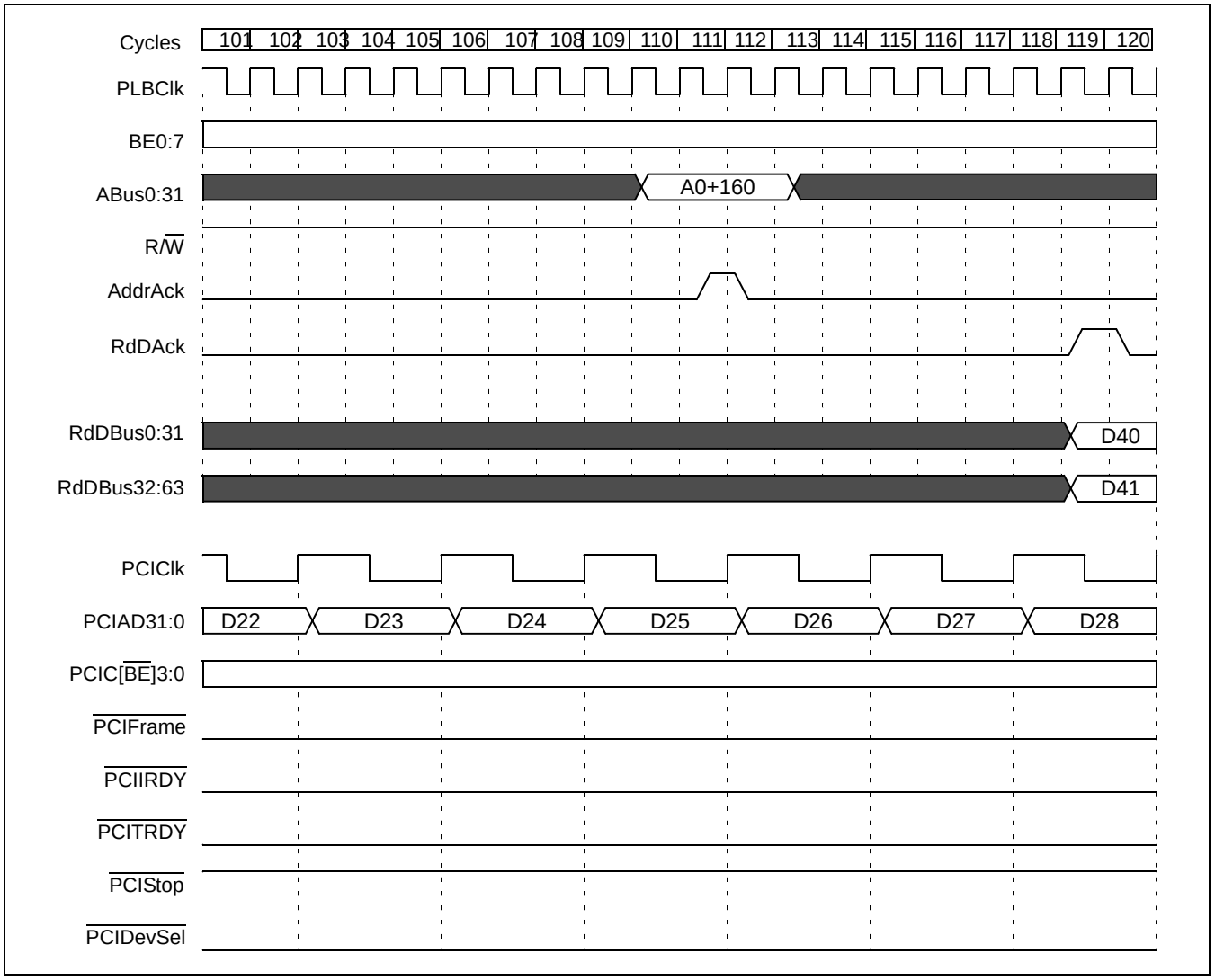


Figure 21-89. PCI Master Burst Read From SDRAM (Continued)

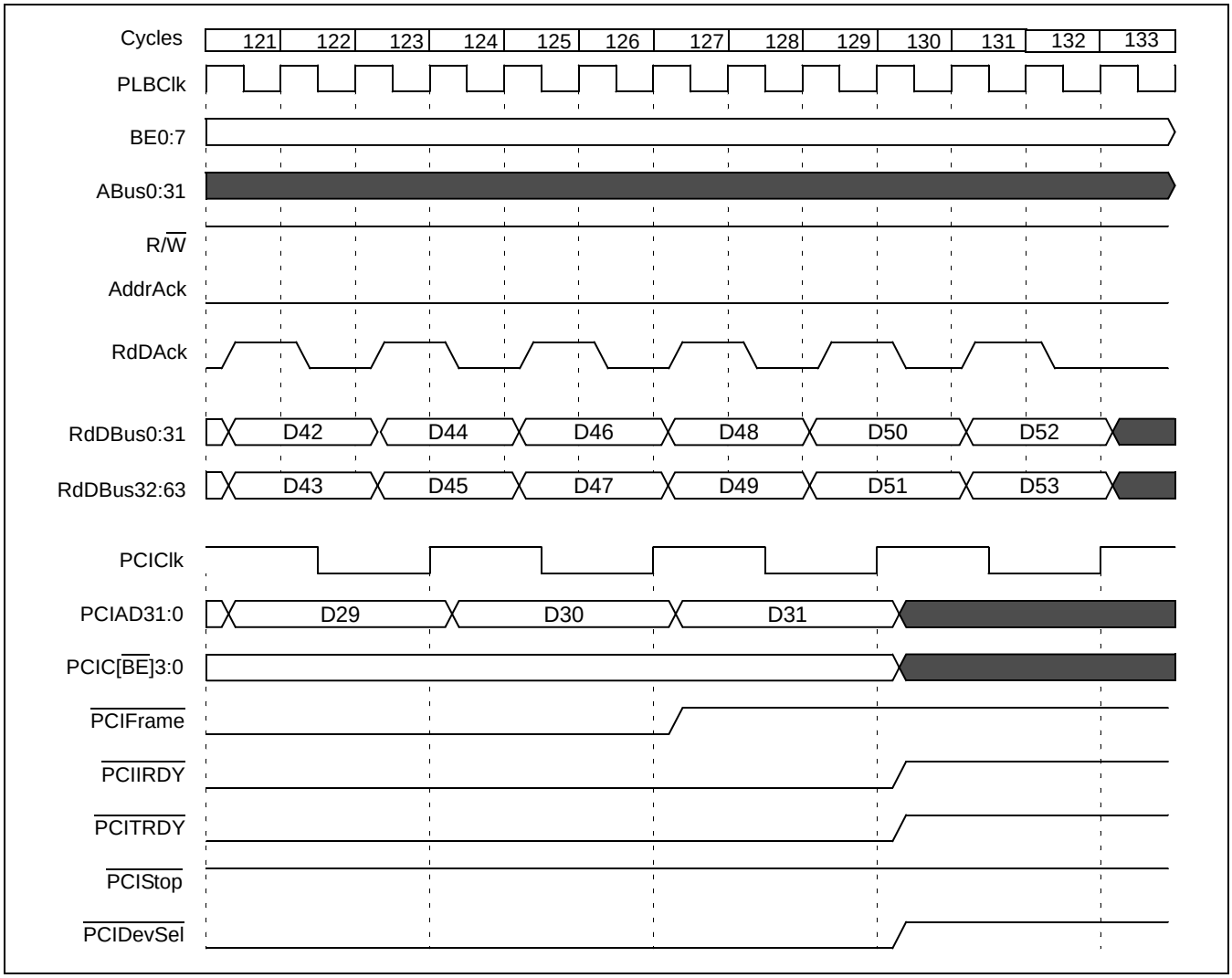


Figure 21-90. PCI Master Burst Write To SDRAM

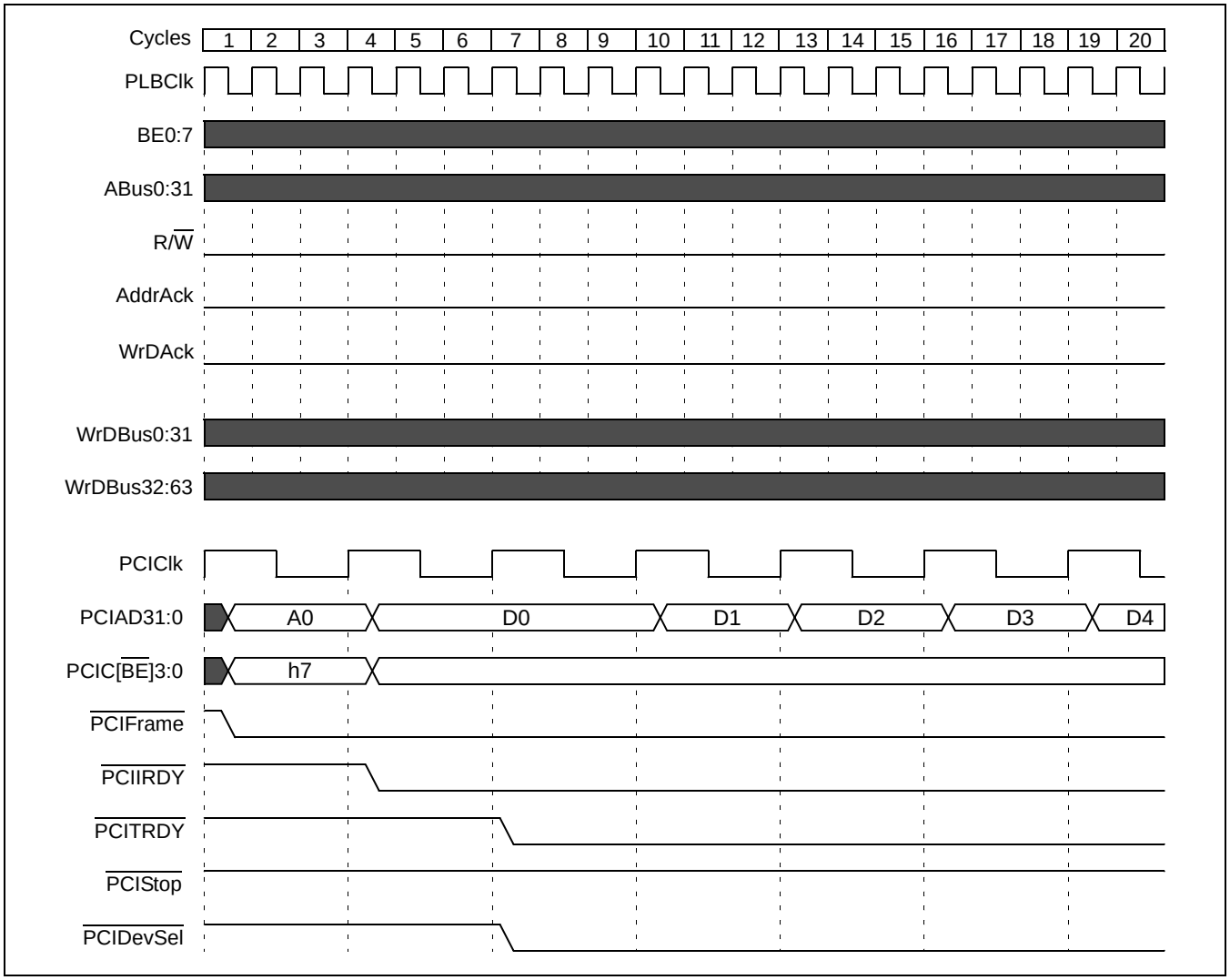


Figure 21-91. PCI Master Burst Write To SDRAM (Continued)

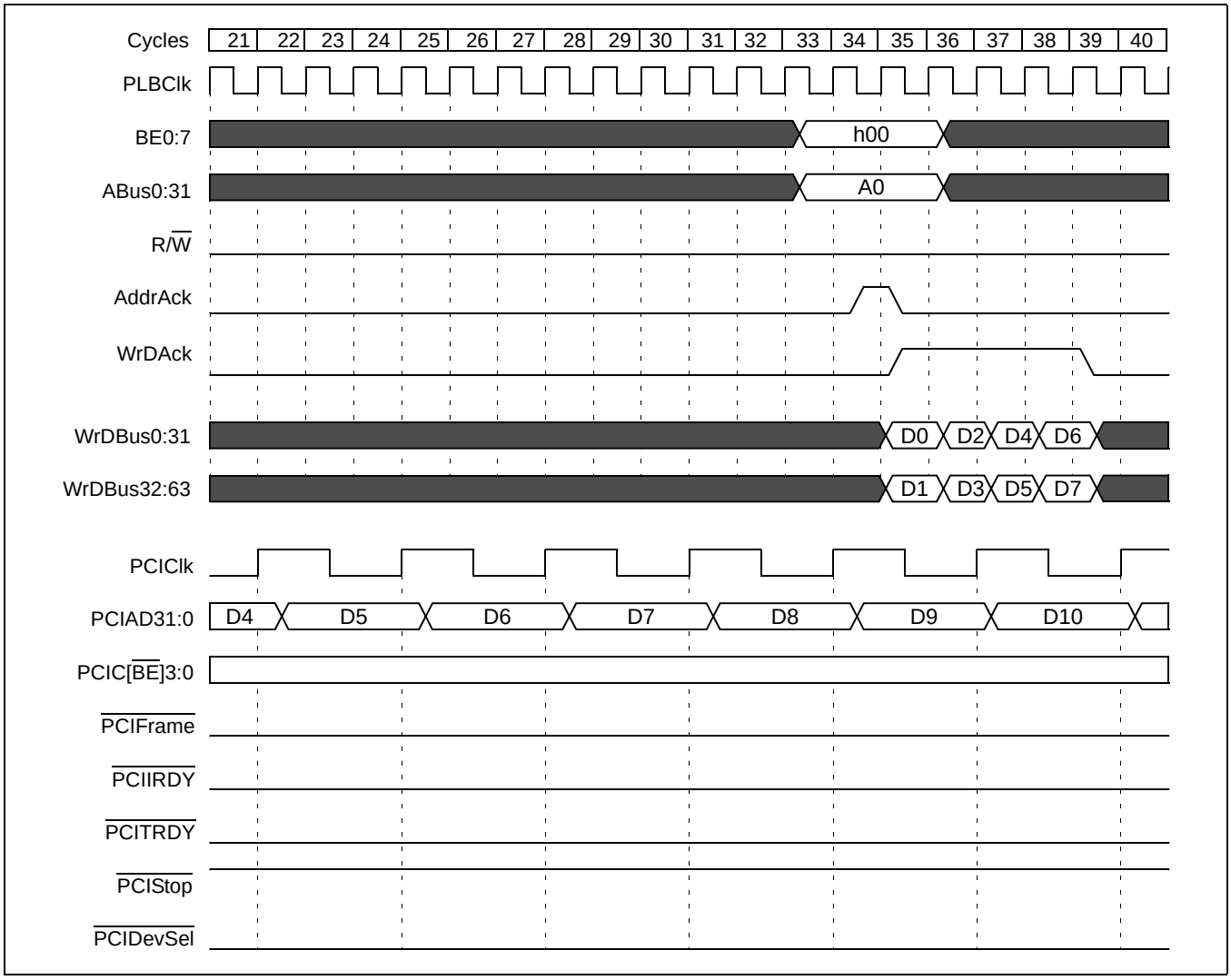


Figure 21-92. PCI Master Burst Write To SDRAM (Continued)

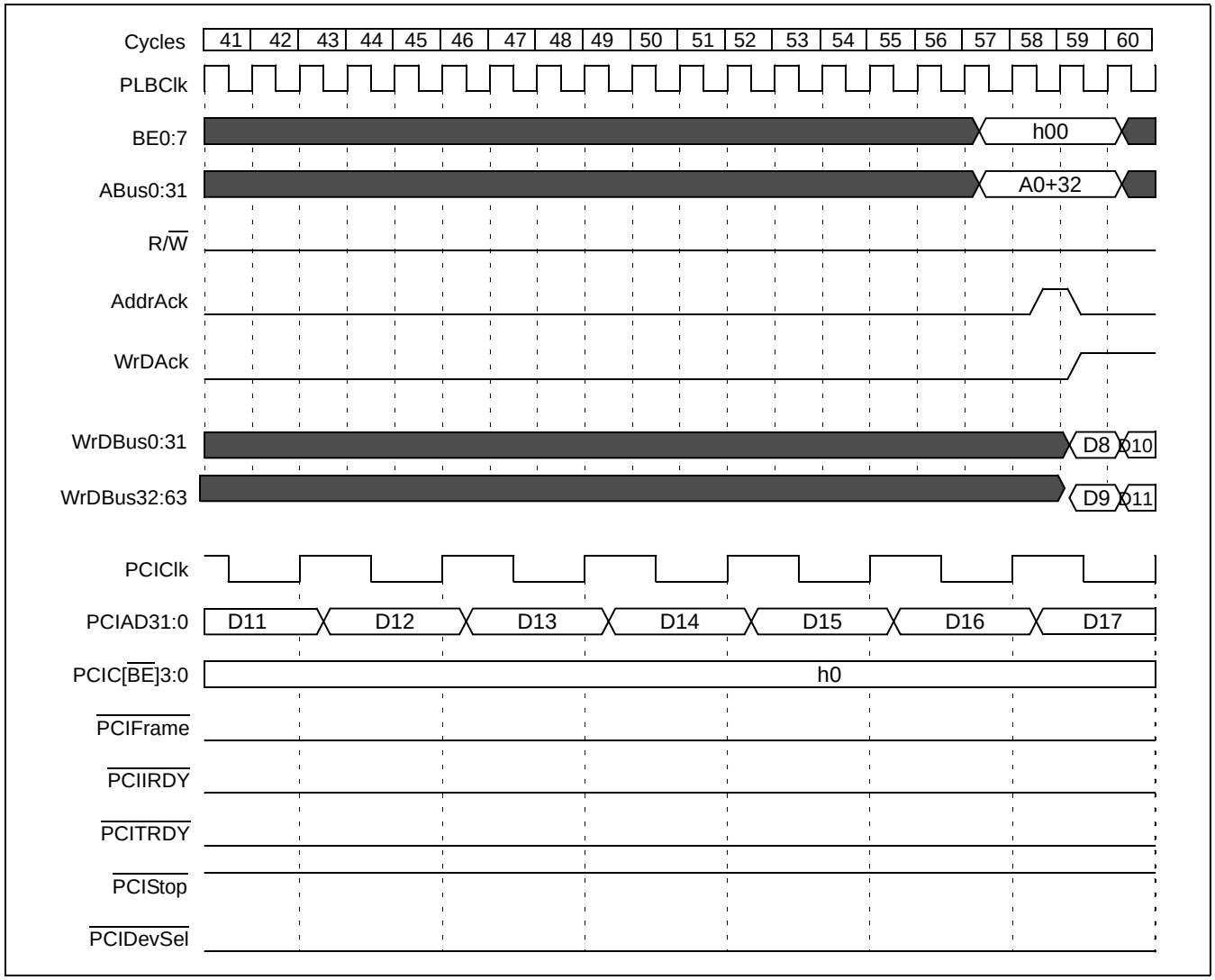


Figure 21-93. PCI Master Burst Write To SDRAM (Continued)

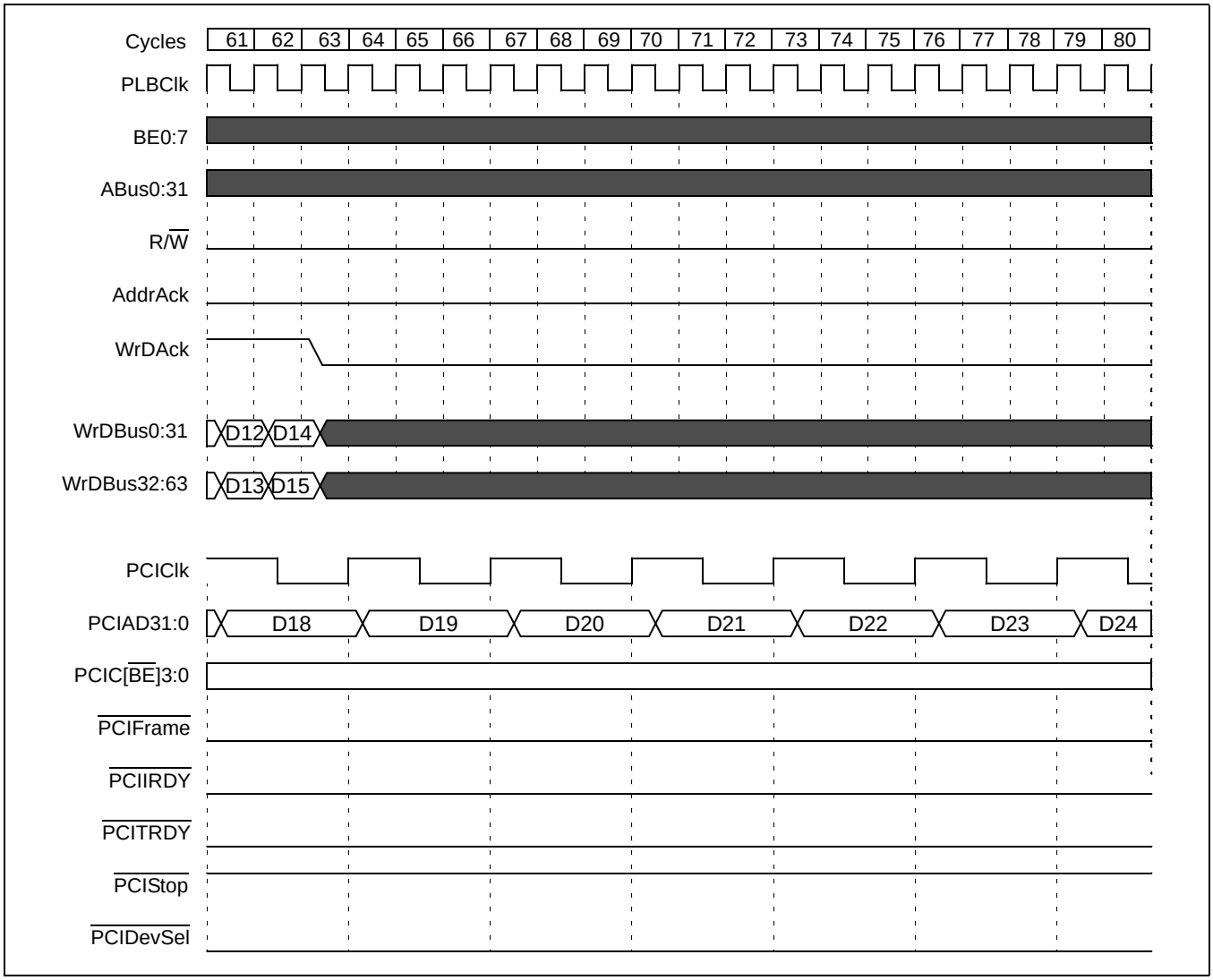


Figure 21-94. PCI Master Burst Write To SDRAM (Continued)

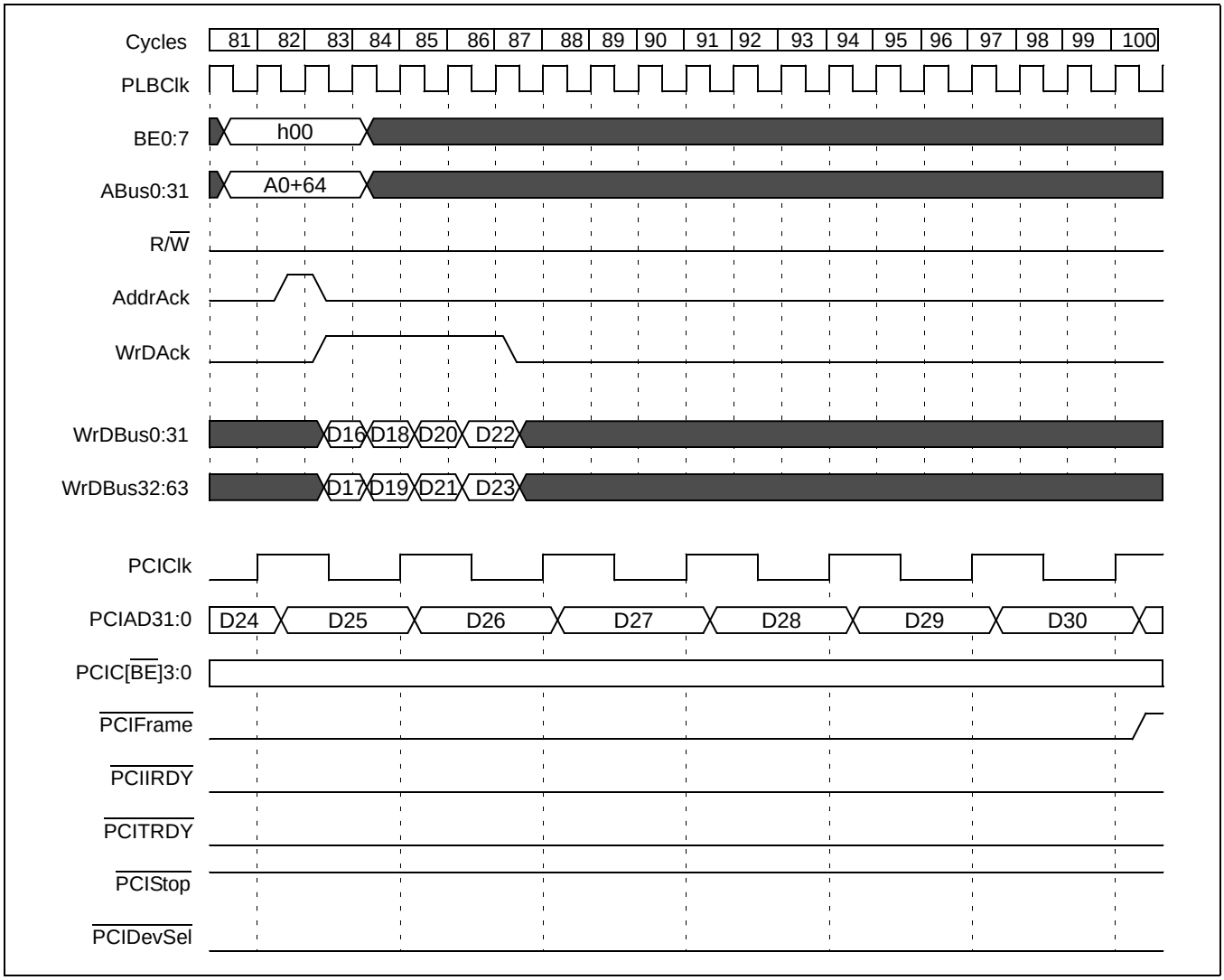


Figure 21-95. PCI Master Burst Write To SDRAM (Continued)

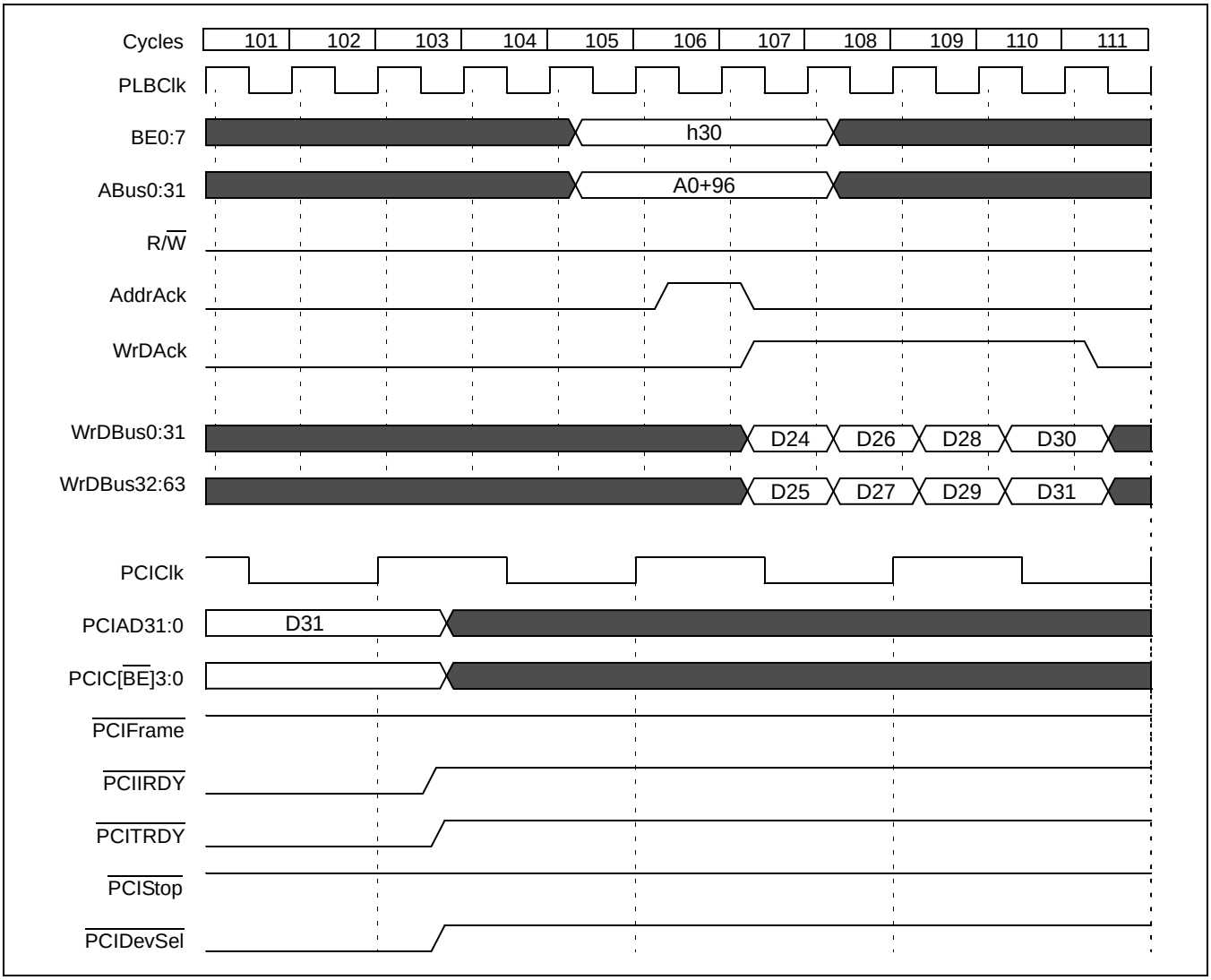


Figure 21-96. CPU Read From PCI Memory Slave, Nonprefetching

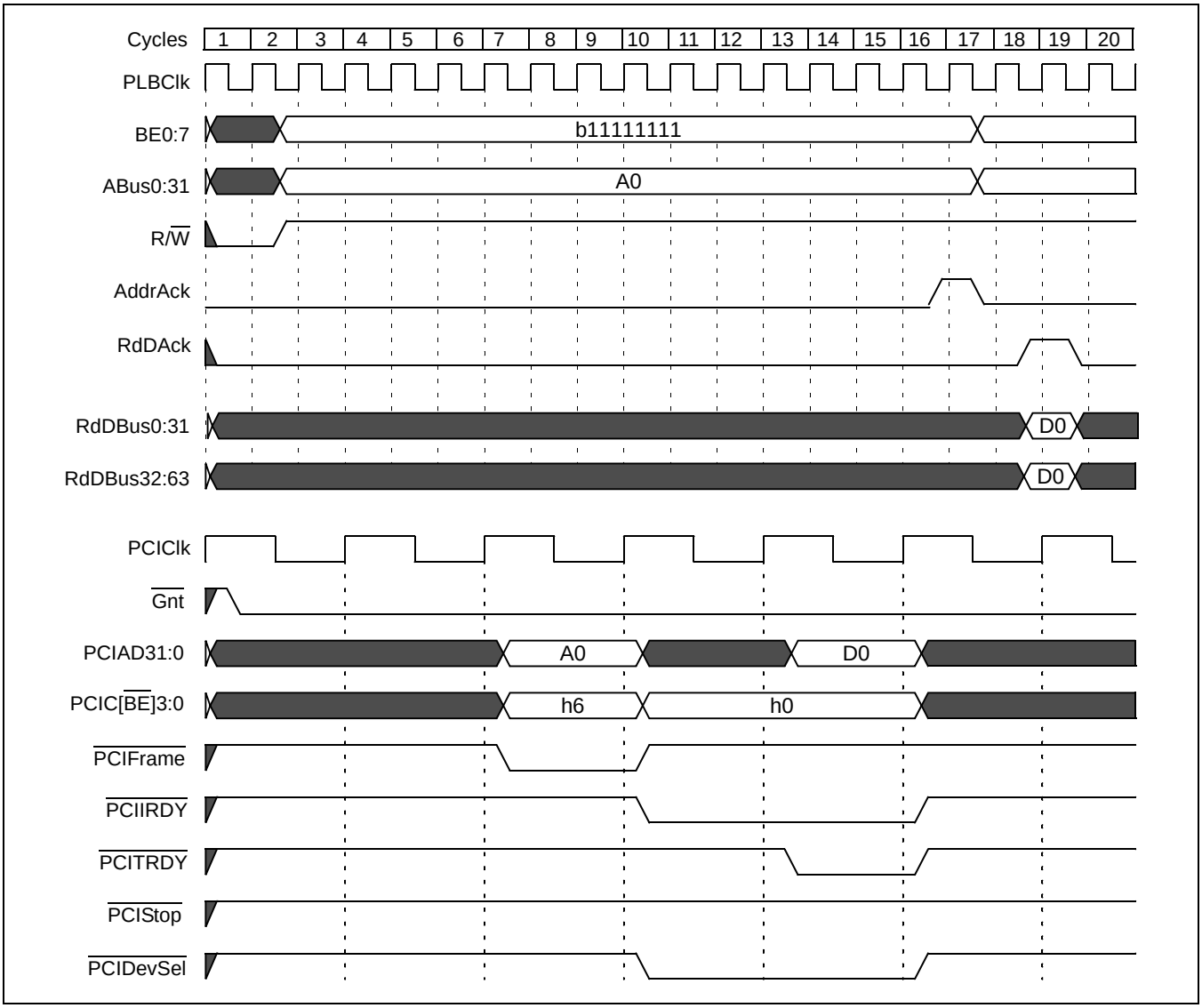


Figure 21-97. CPU Read From PCI Memory Slave, Nonprefetching (Continued)

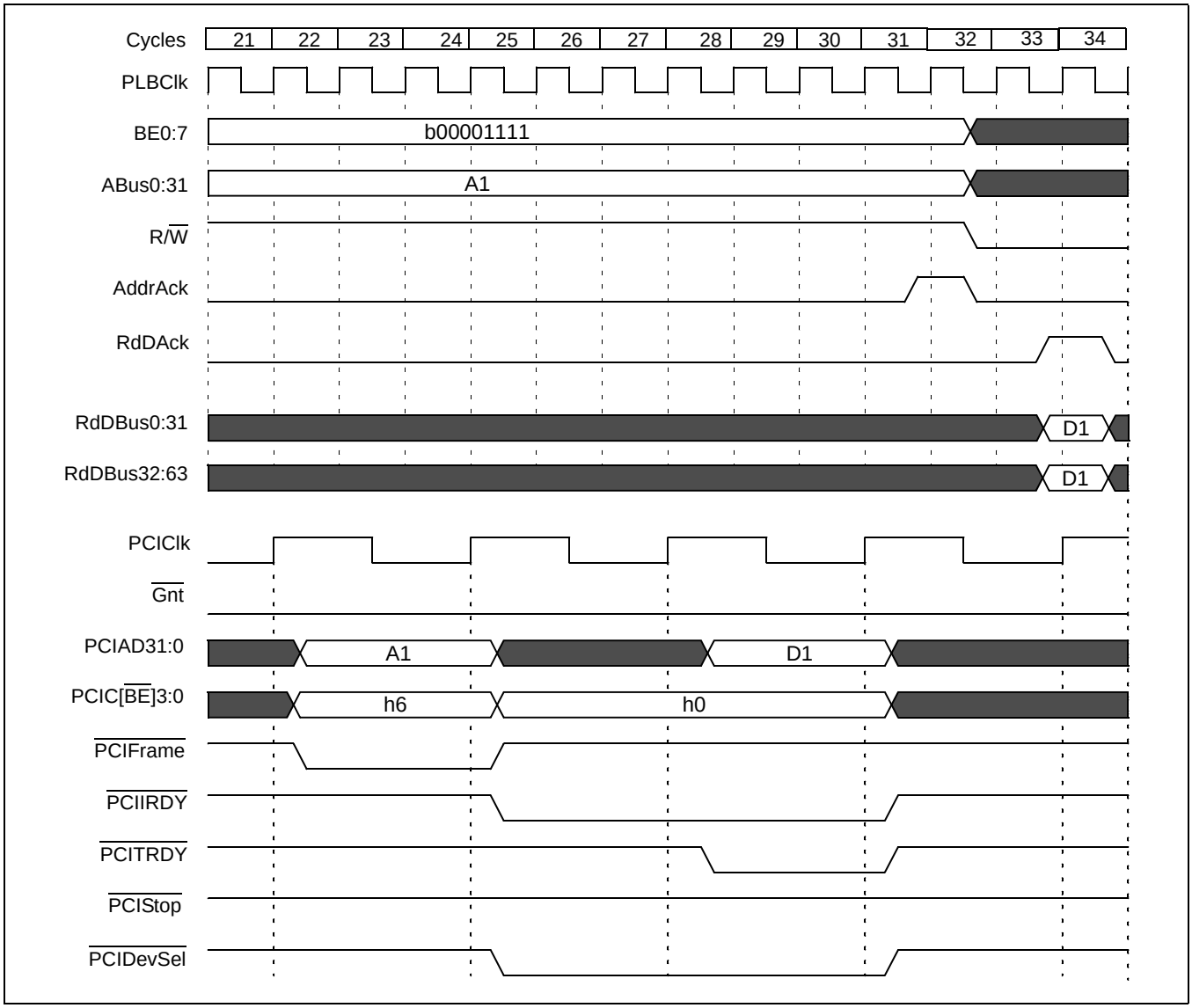


Figure 21-98. CPU Read From PCI Memory Slave, Prefetching

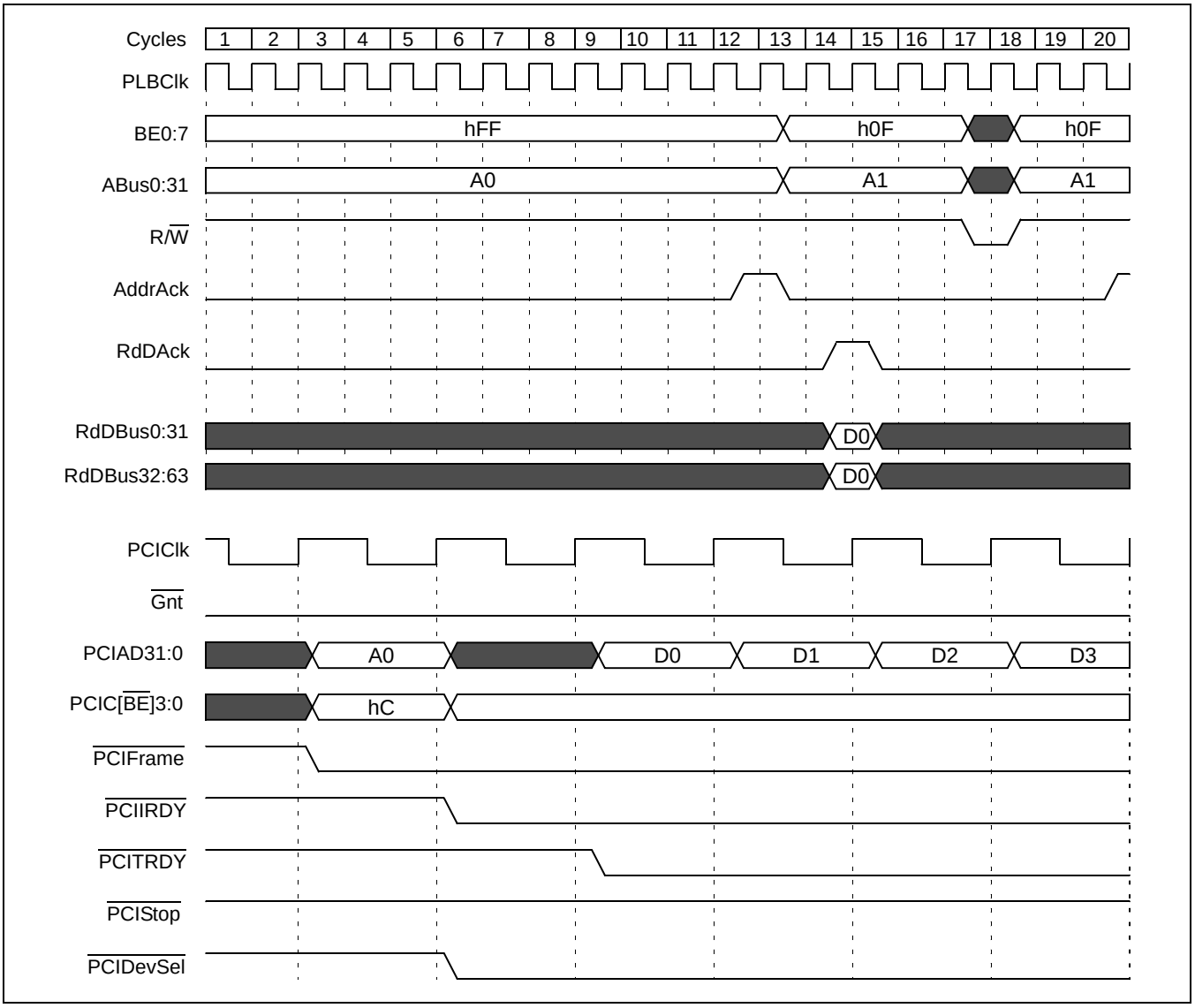


Figure 21-99. CPU Read From PCI Memory Slave, Prefetching (Continued)

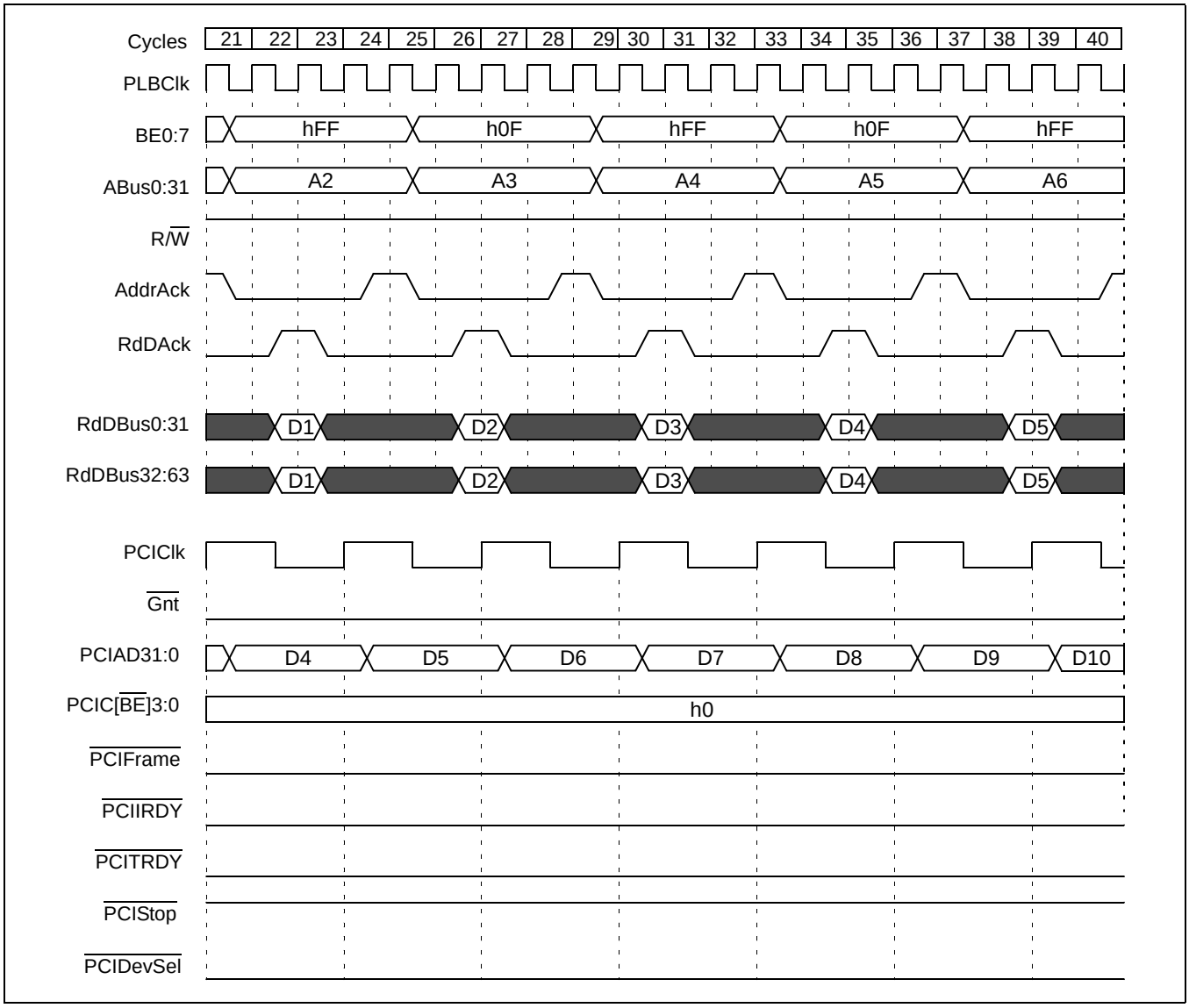


Figure 21-100. CPU Read From PCI Memory Slave, Prefetching (Continued)

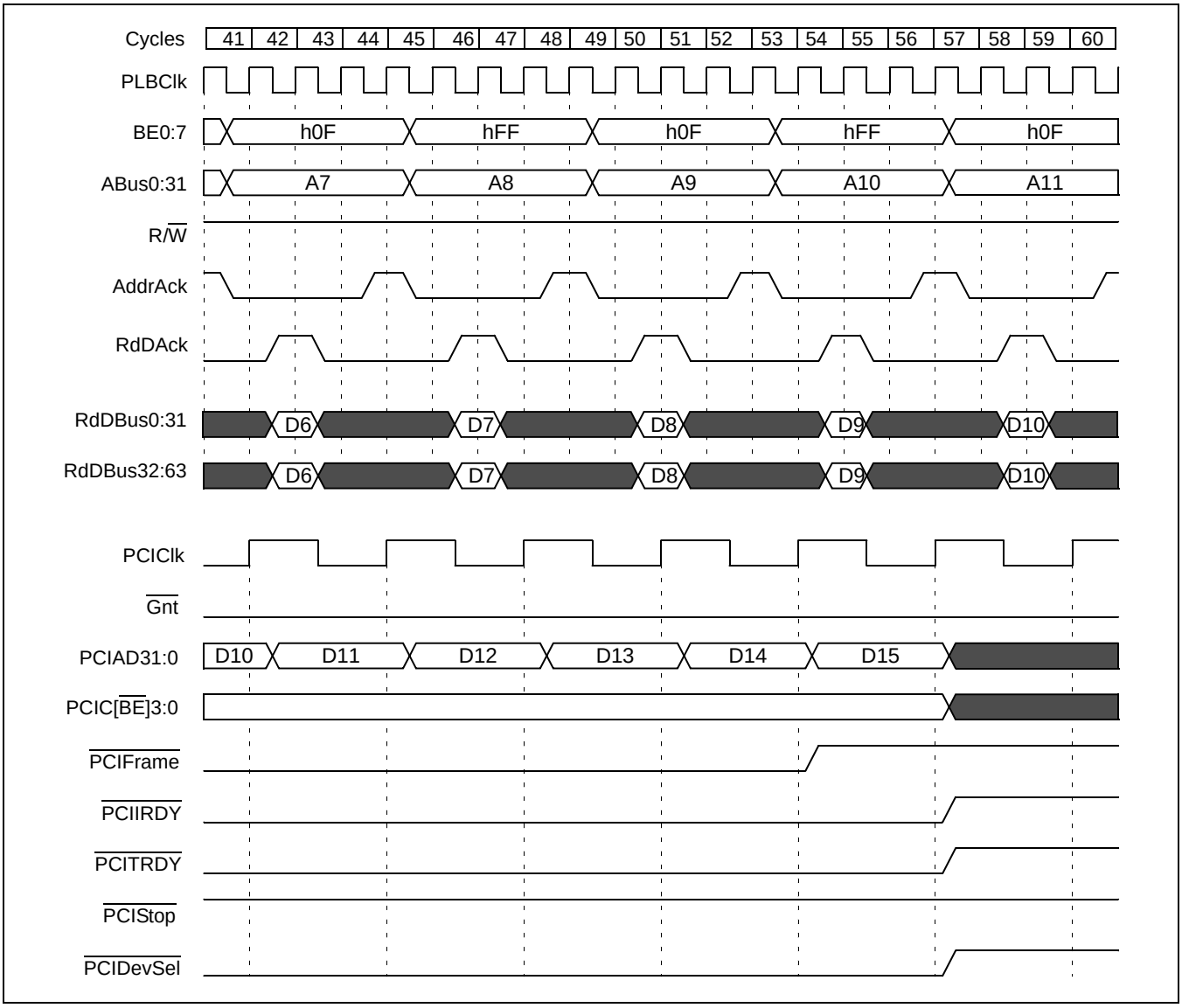


Figure 21-101. CPU Read From PCI Memory Slave, Prefetching (Continued)

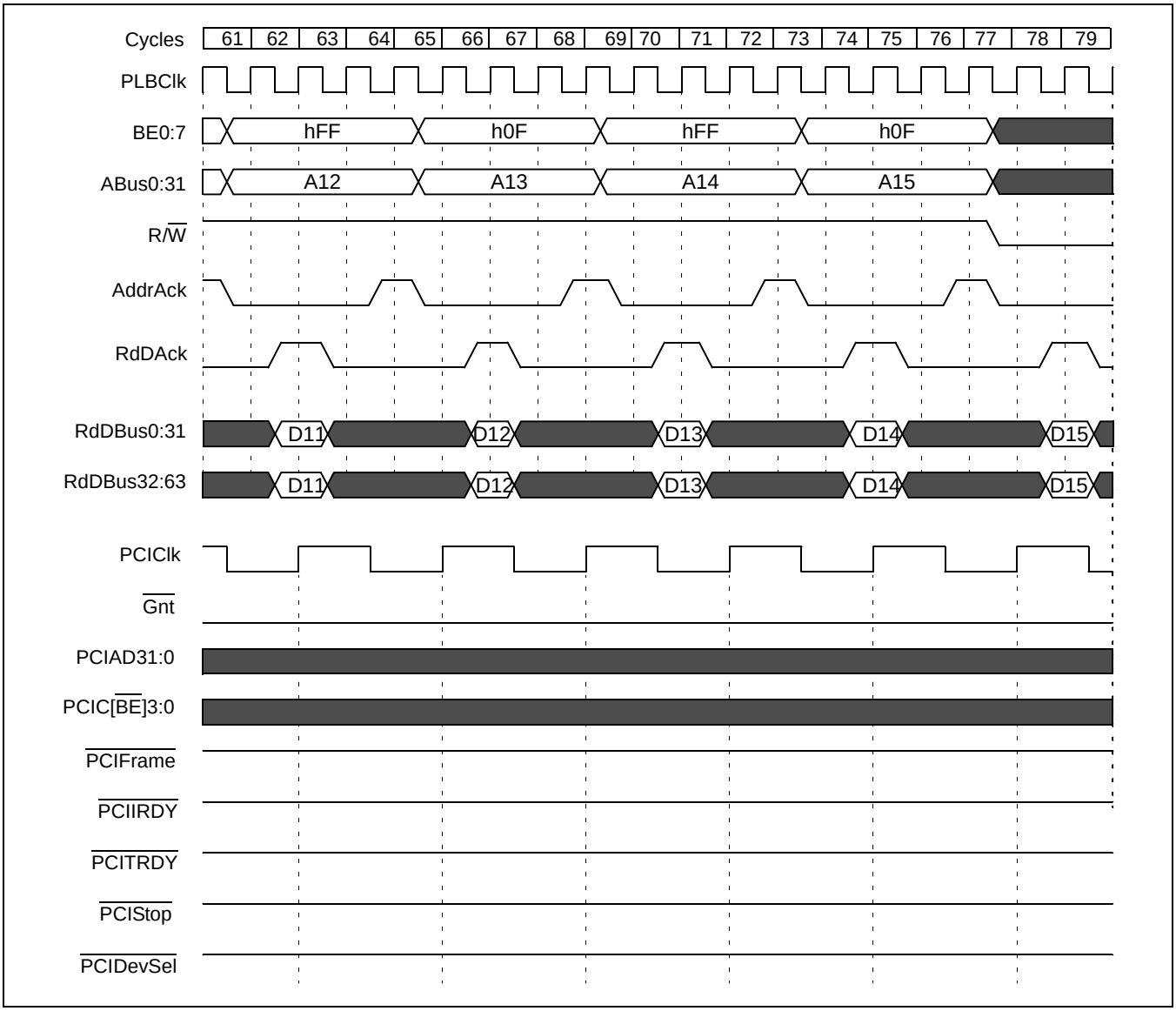


Figure 21-102. CPU Write To PCI Memory Slave

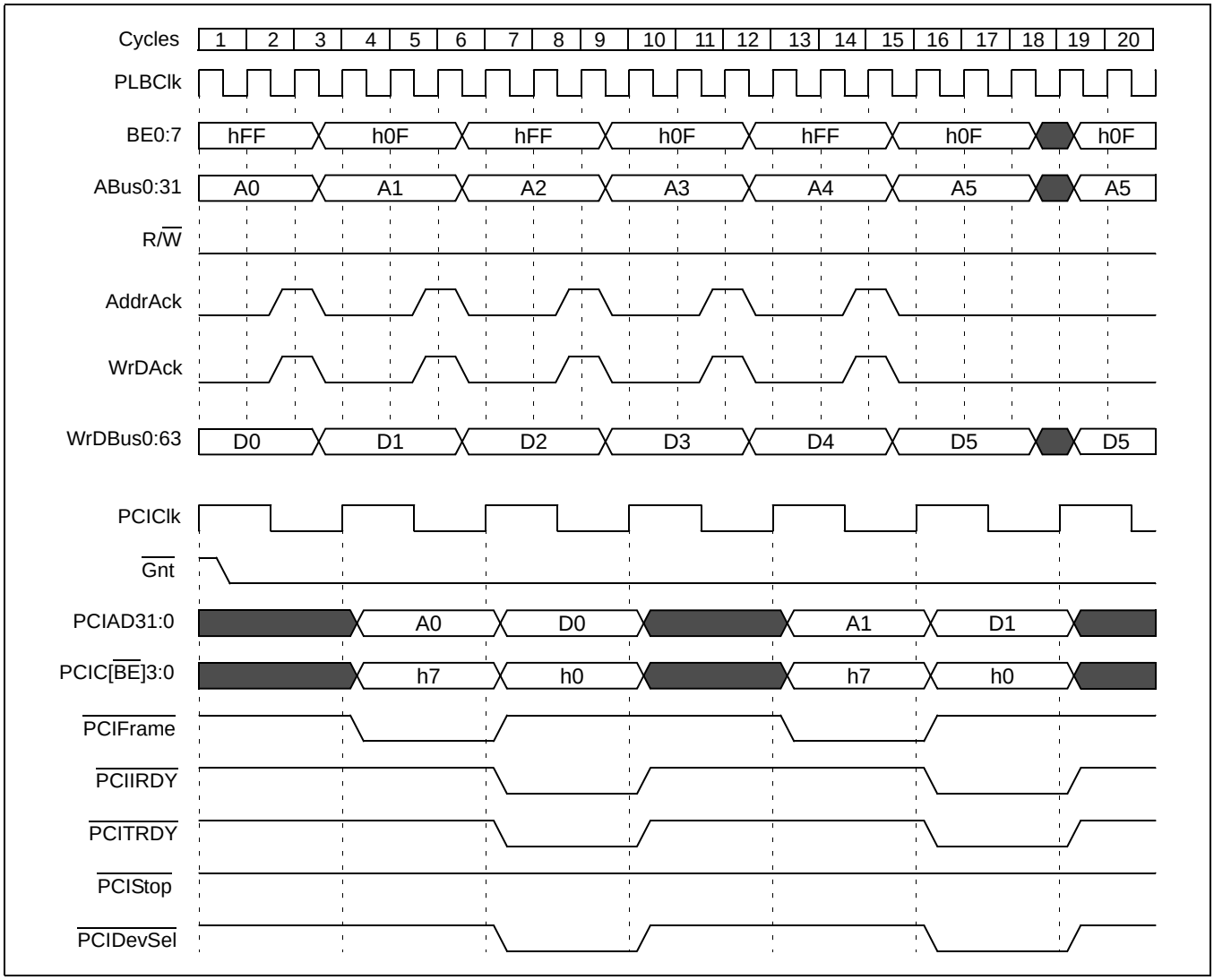


Figure 21-103. CPU Write To PCI Memory Slave (Continued)

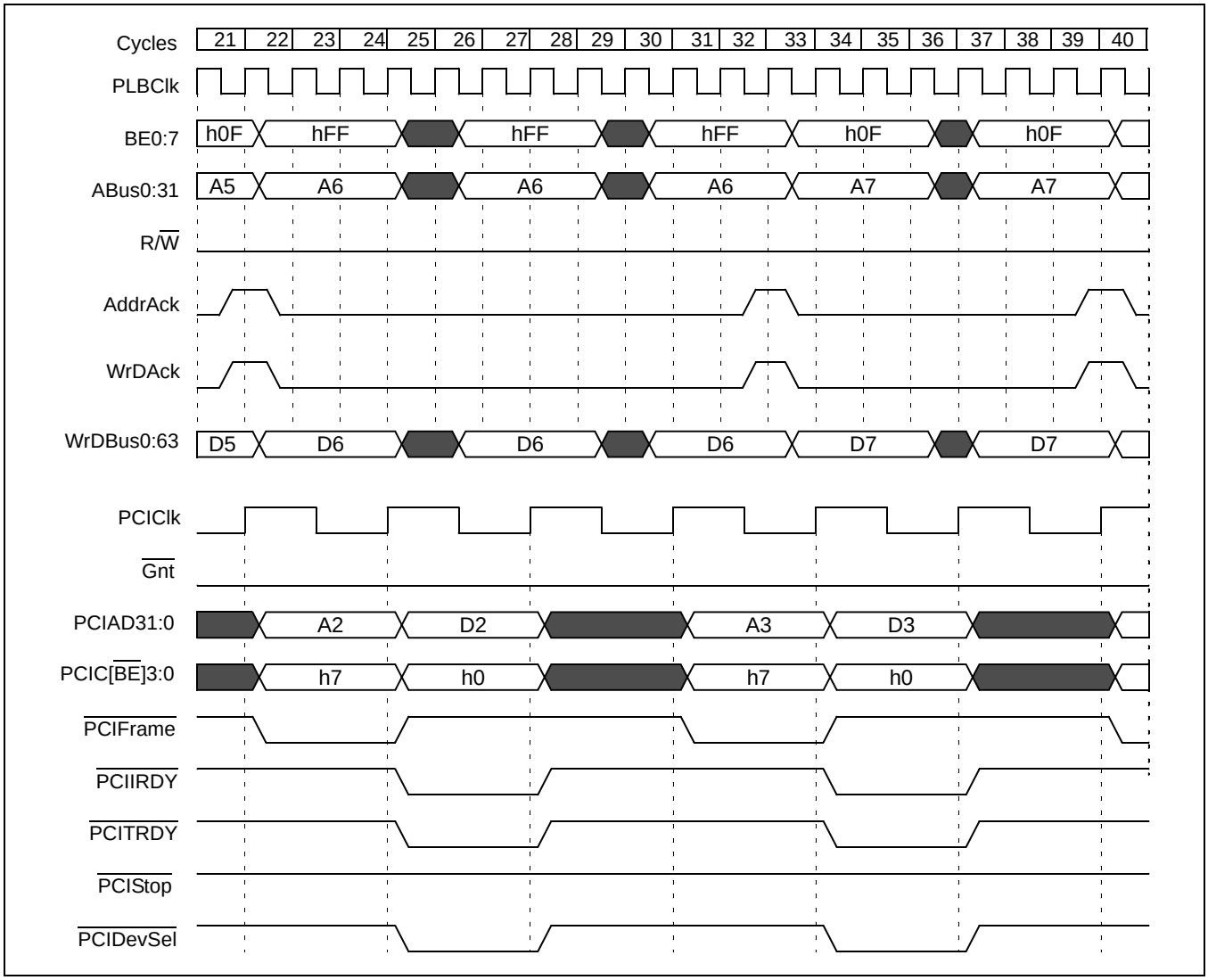


Figure 21-104. CPU Write To PCI Memory Slave (Continued)

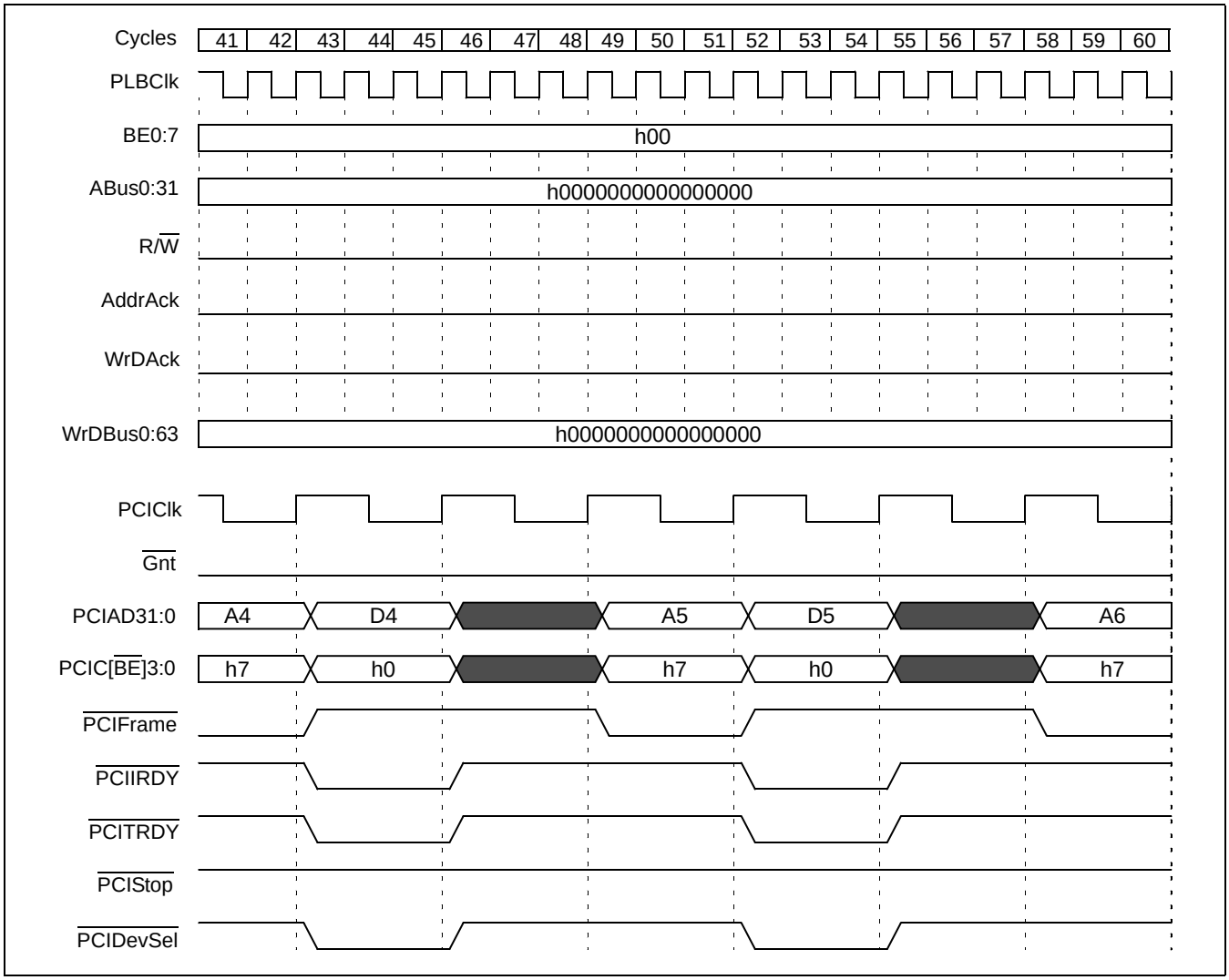


Figure 21-105. CPU Write To PCI Memory Slave (Continued)

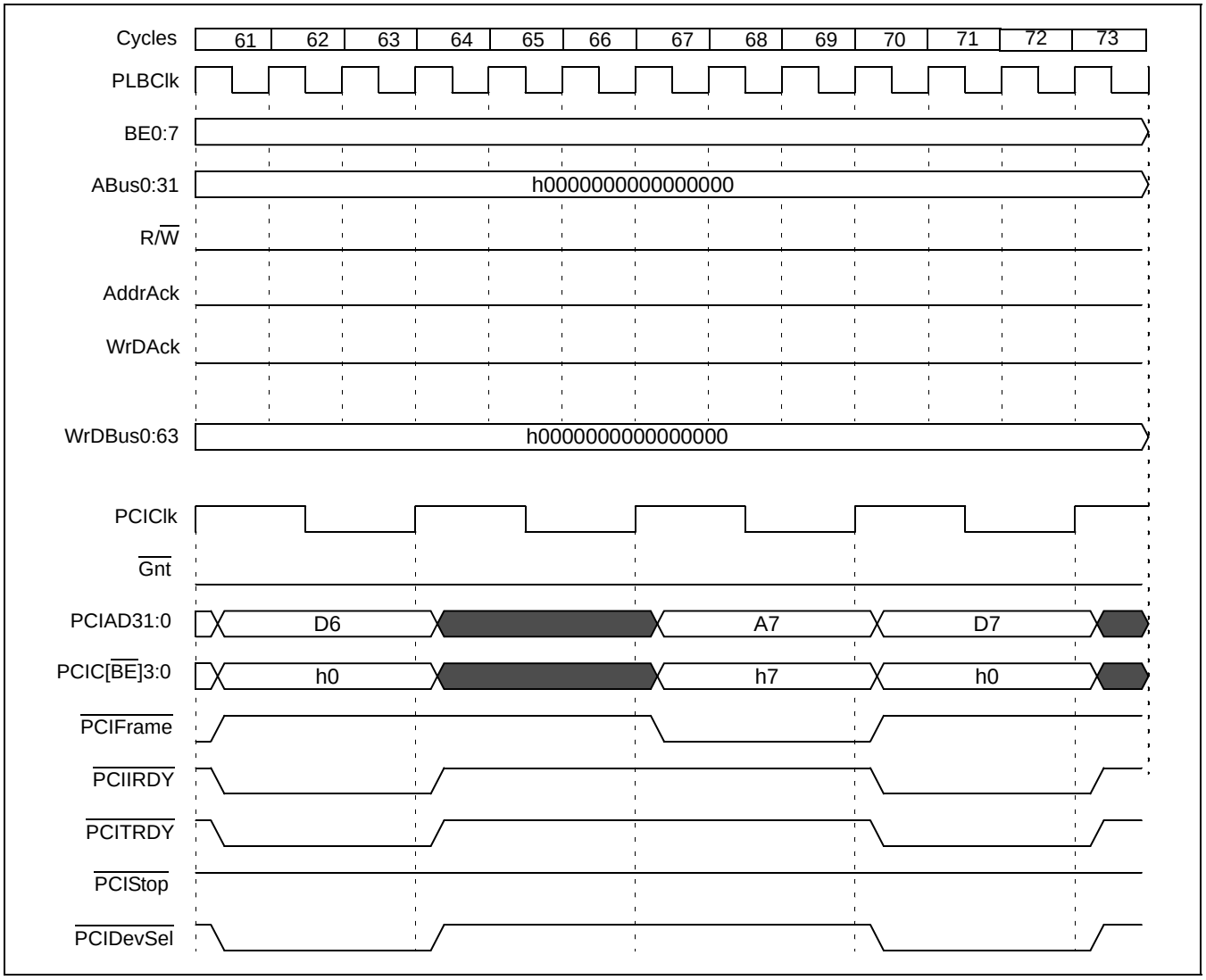


Figure 21-106. PCI Memory To SDRAM DMA Transfer

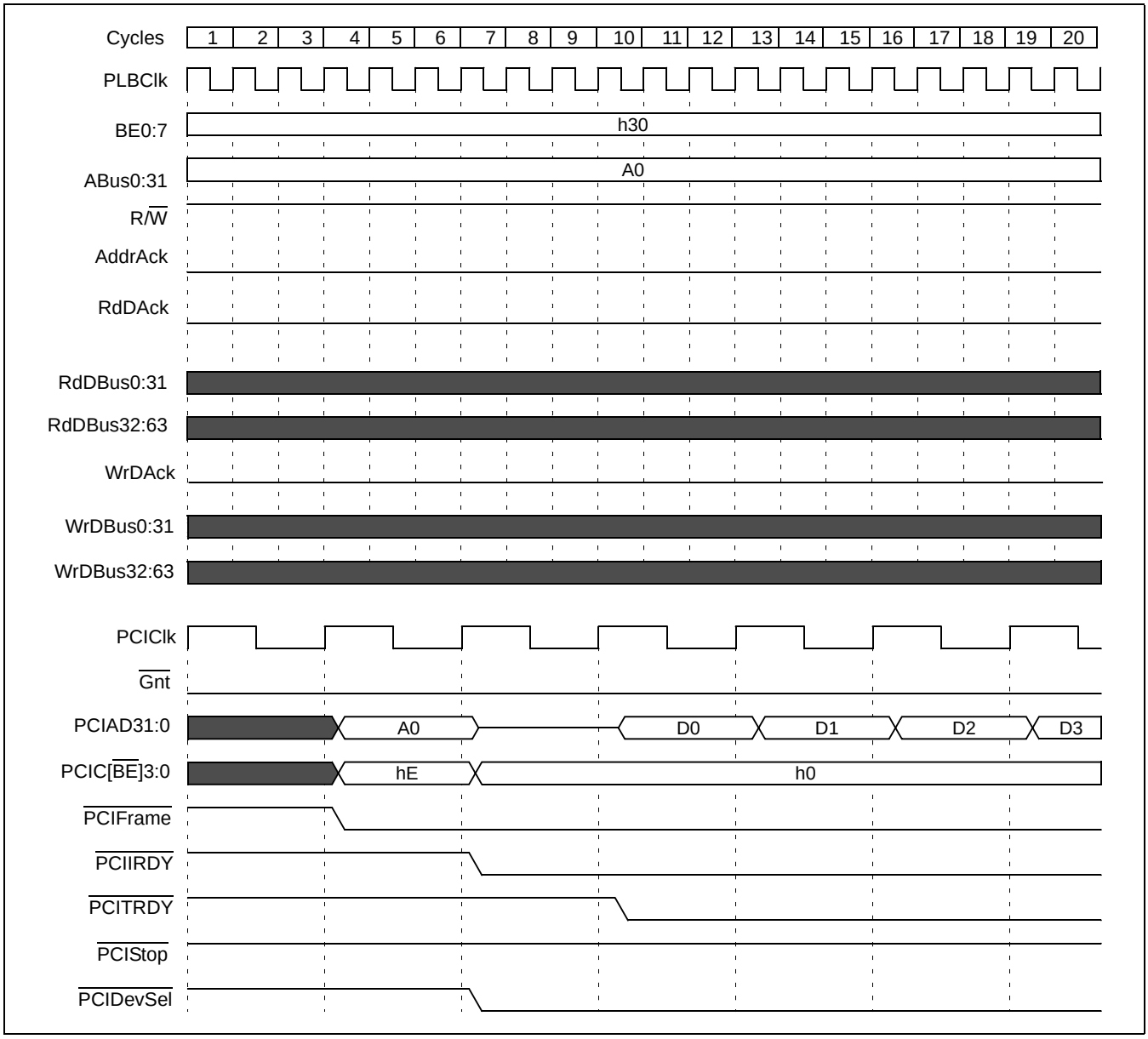


Figure 21-107. PCI Memory To SDRAM DMA Transfer (Continued)

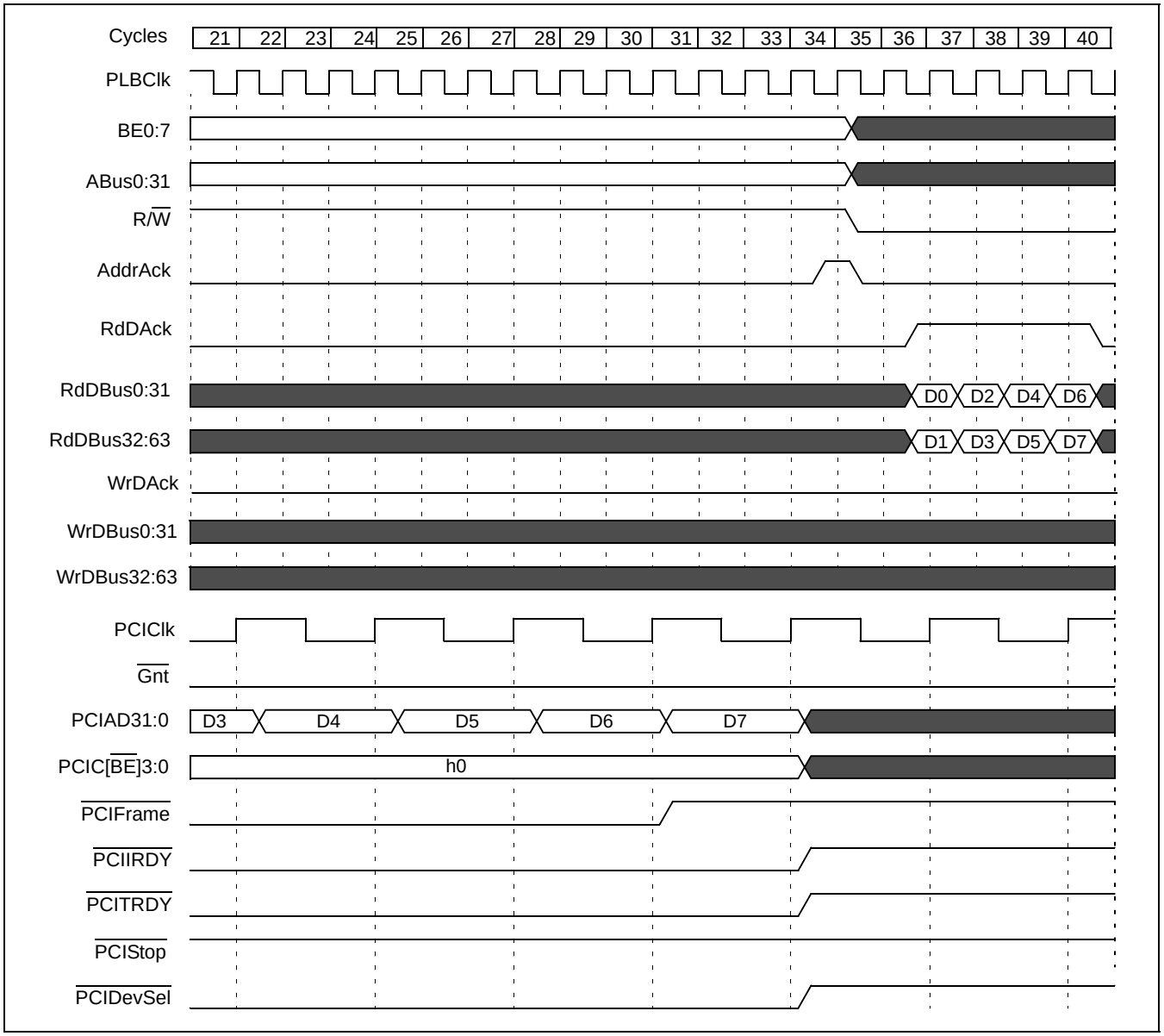


Figure 21-108. PCI Memory To SDRAM DMA Transfer (Continued)

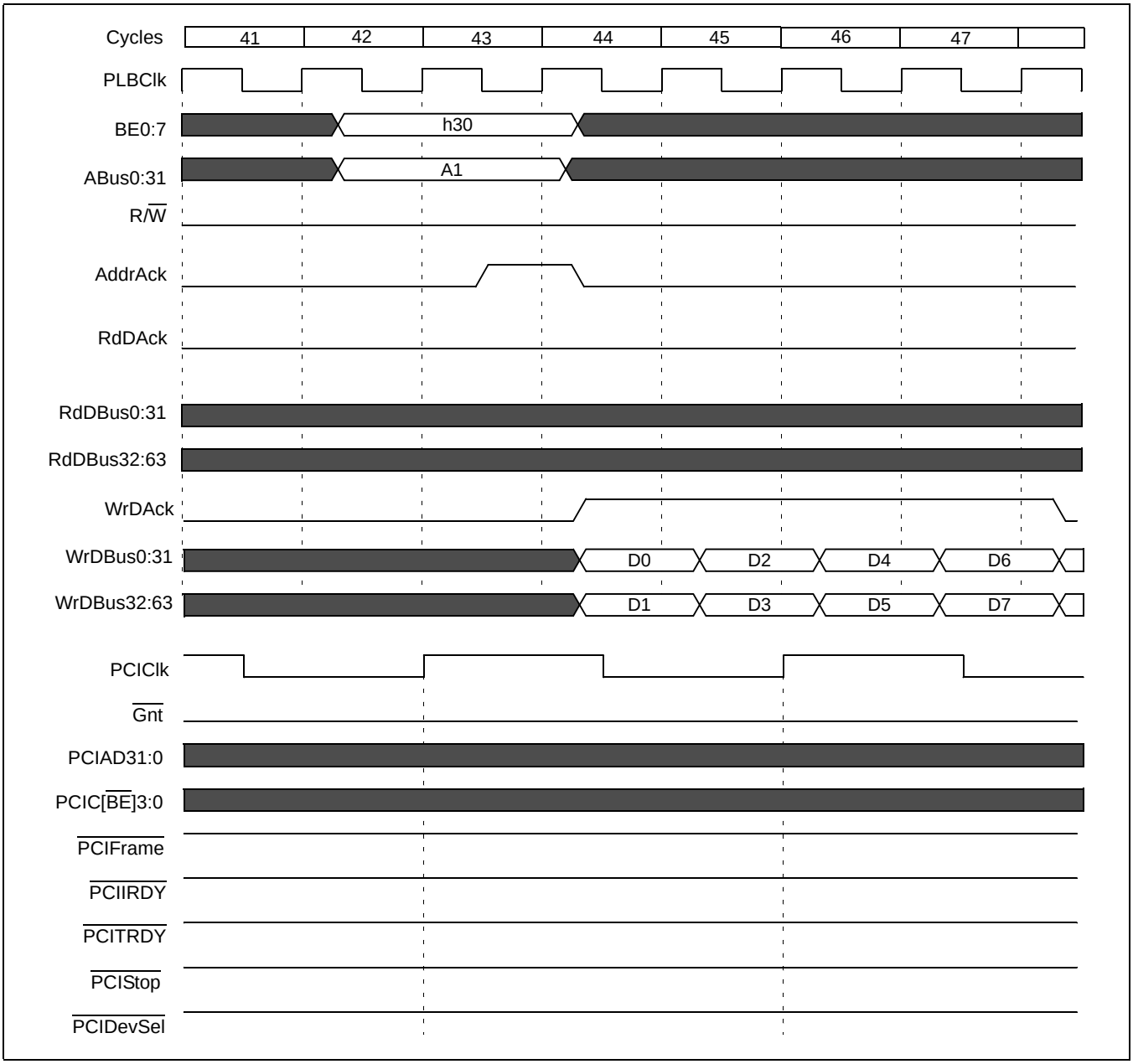


Figure 21-109. SDRAM To PCI Memory DMA Transfer

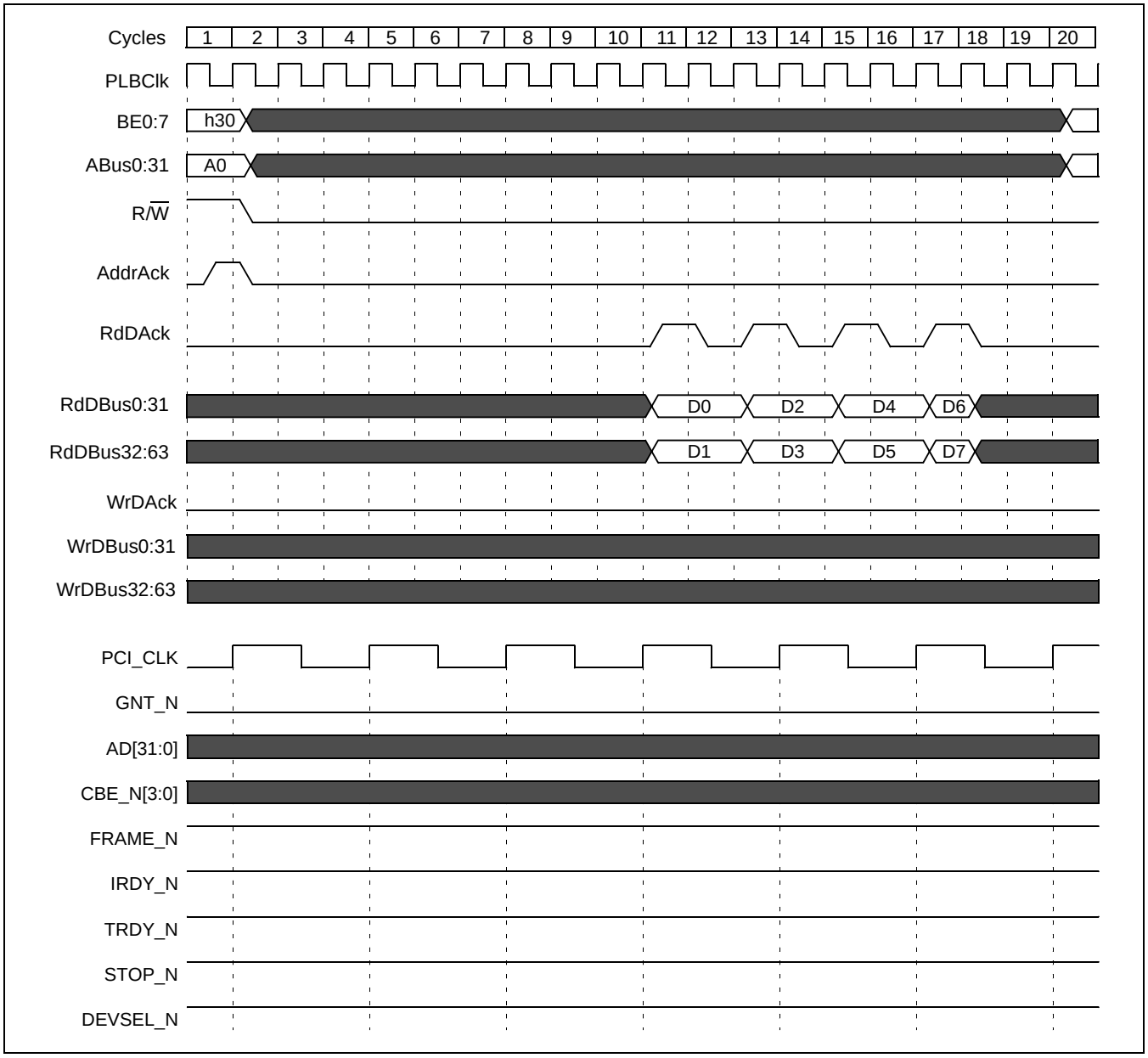


Figure 21-110. SDRAM To PCI Memory DMA Transfer (Continued)

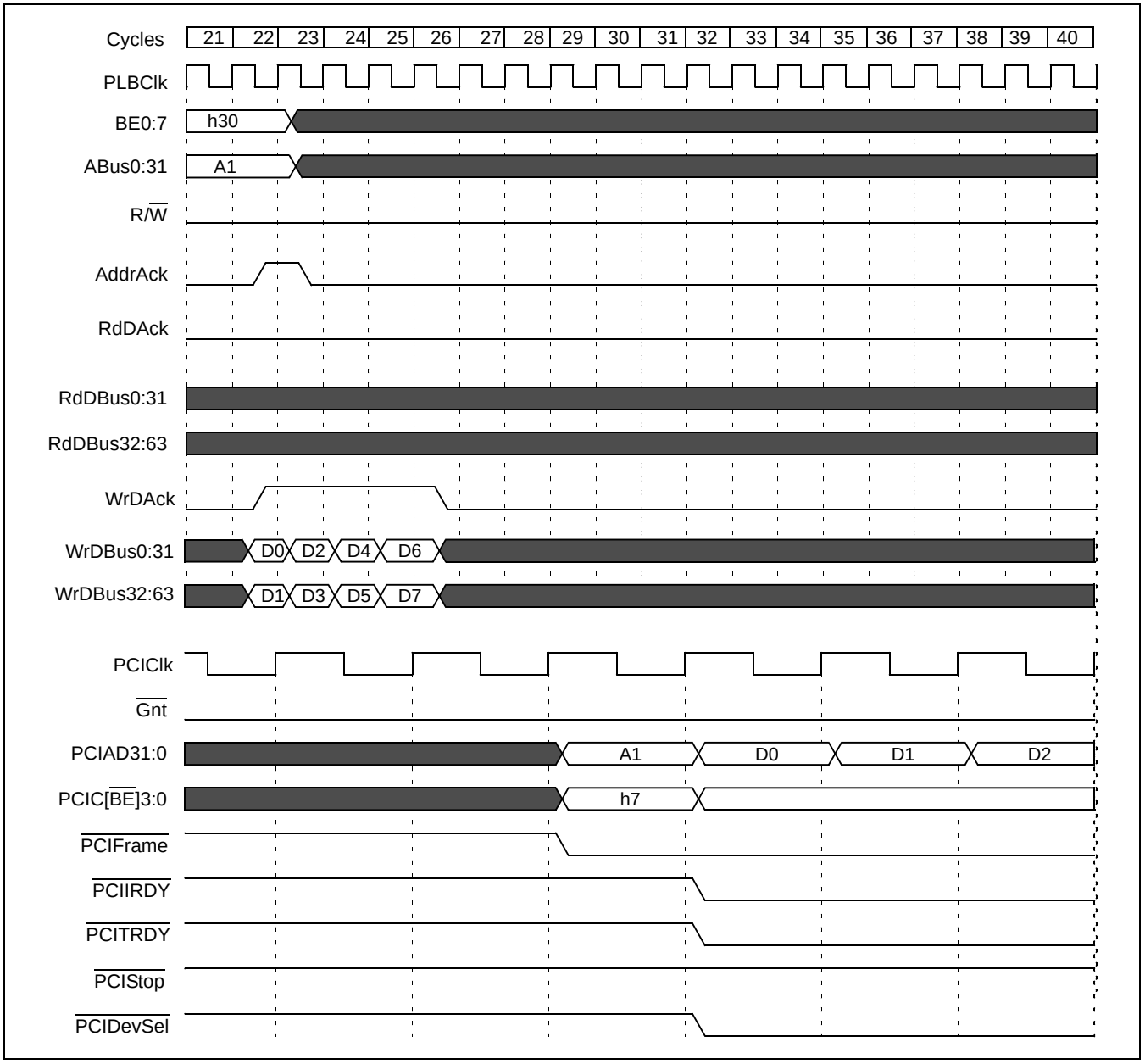
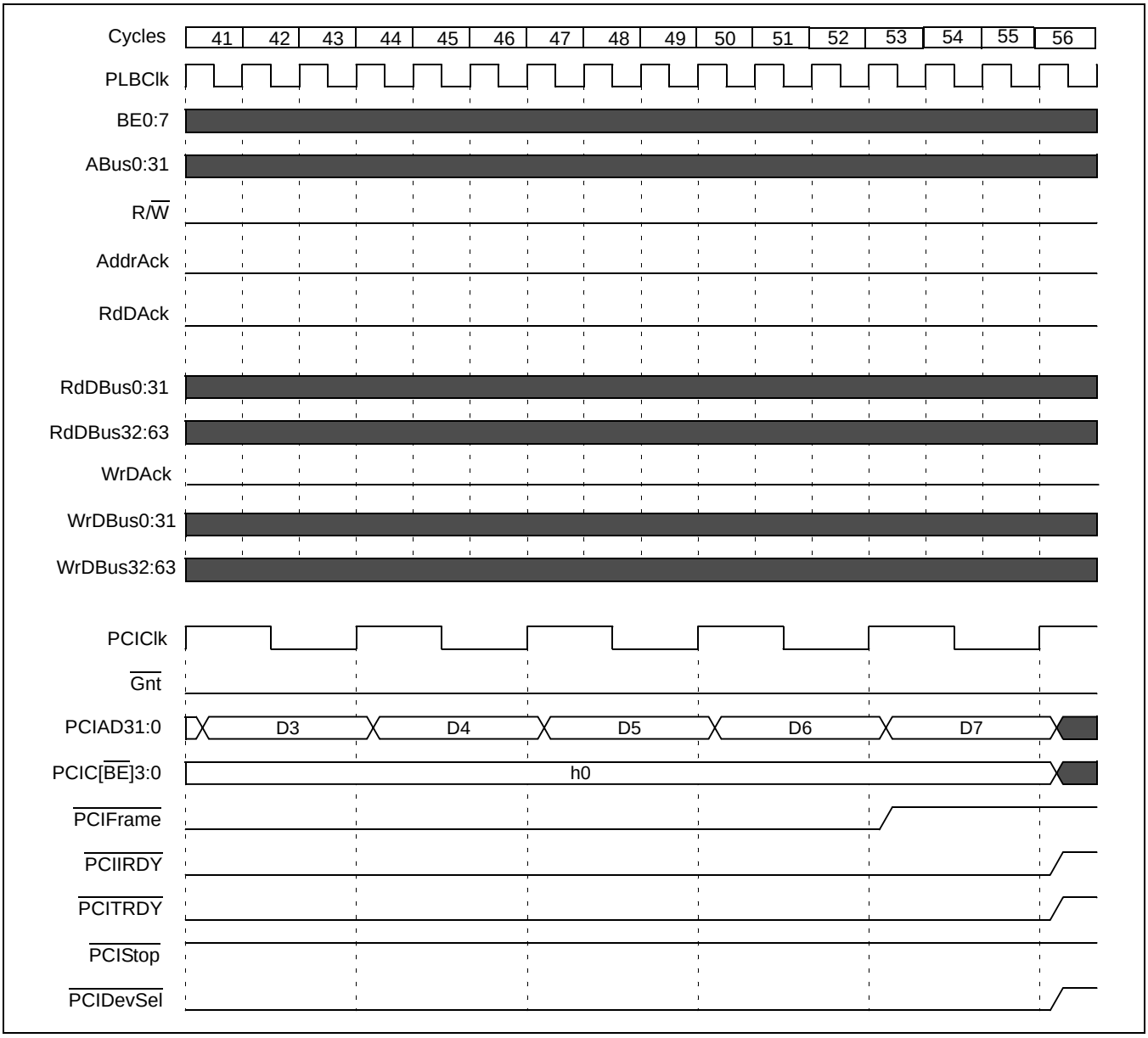


Figure 21-111. SDRAM To PCI Memory DMA Transfer (Continued)



22. External Bus Controller

The External Bus Controller (EBC) provides direct attachment for most SRAM, Flash memory, and peripheral devices. The interface minimizes the amount of external glue logic needed to communicate with memory and peripheral devices, reducing embedded system device count, circuit board area, and cost.

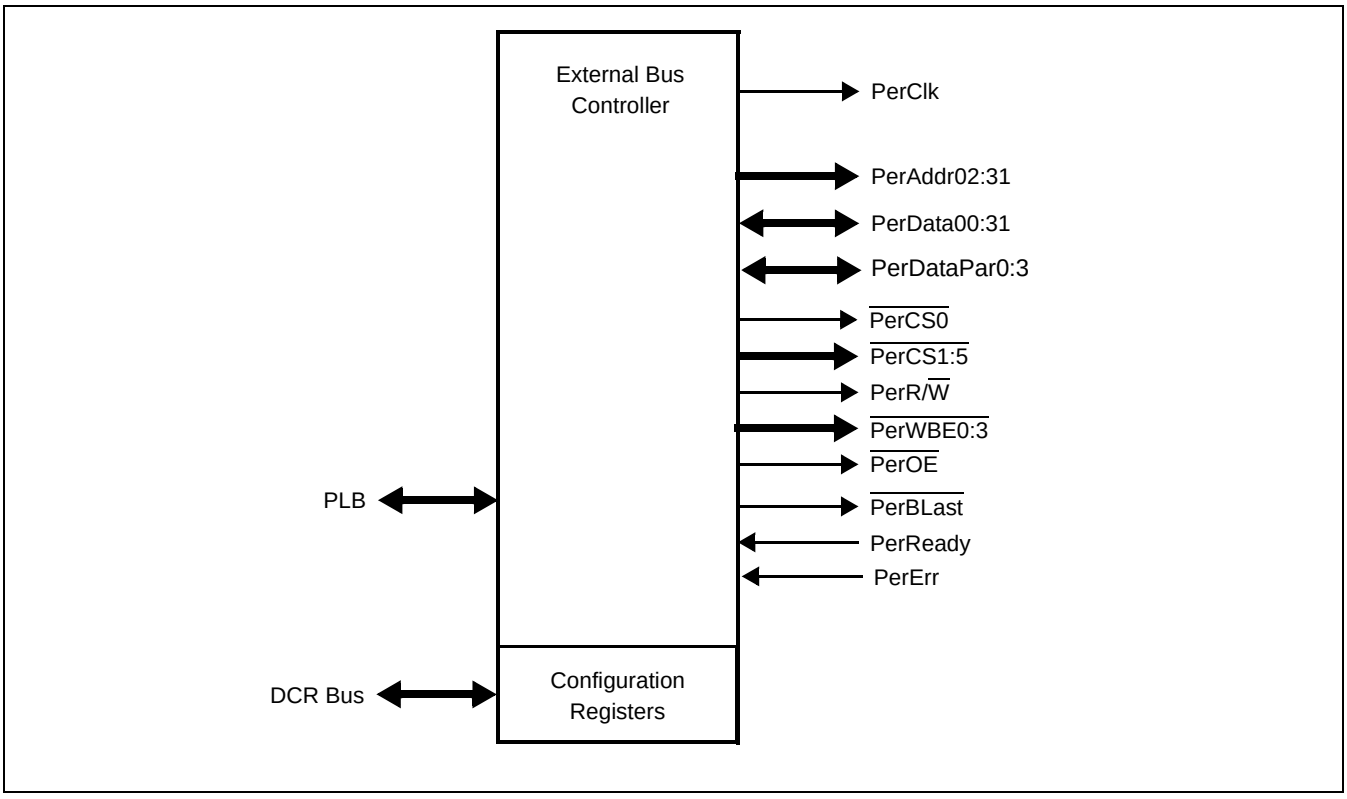
To eliminate off-chip address decoding, the EBC provides six programmable chip selects that enable system designers to locate memory and peripherals within the PPC440EPx/GRx memory map. Chip select, data bus, and associated control signal timings are programmable for both single and burst transfers. For peripherals with variable timing requirements, the EBC supports device-paced transfers with optional bus-timeout. System design is further simplified through dynamic bus sizing that supports seamlessly attaching 8-, 16-, and 32-bit wide memories and peripherals. Whenever a size mismatch exists between a read or write operation and the externally attached device, the EBC automatically packs or unpacks data as appropriate.

Data parity is supported (one parity bit per data byte).

22.1 Interface Signals

The *Figure 22-1* illustrates the signal I/O between the EBC and the external peripheral bus.

Figure 22-1. EBC Signals



The following table describes signal usage and state during and after chip and system resets.

Table 22-1. EBC Signal Usage

Signal	Usage
PerClk	Peripheral bus clock. During an EBC transfer all EBC signal transitions and data sampling occurs synchronous to PerClk.
PerAddr02:31	Peripheral address bus. PerAddr2 is the most significant bit.
PerData0:31	Peripheral data bus. PerData0 is the most significant bit.
PerDataPar0:3	Peripheral data bus Parity for PerData0:7, PerData8:15, PerData16:23, and PerData24:31
PerCS0:5	Chip selects.
PerR/W	Read not write.
PerWBE0:3	Write byte enables or read/write byte enables.
PerOE	Output enable.
PerBLast	Burst Last. Active during non-burst operations and the last transfer of a burst access.
PerReady	An input to allow external peripherals to perform device-paced transfers.
PerErr	Peripheral data error input. Sampled only during data transfers.

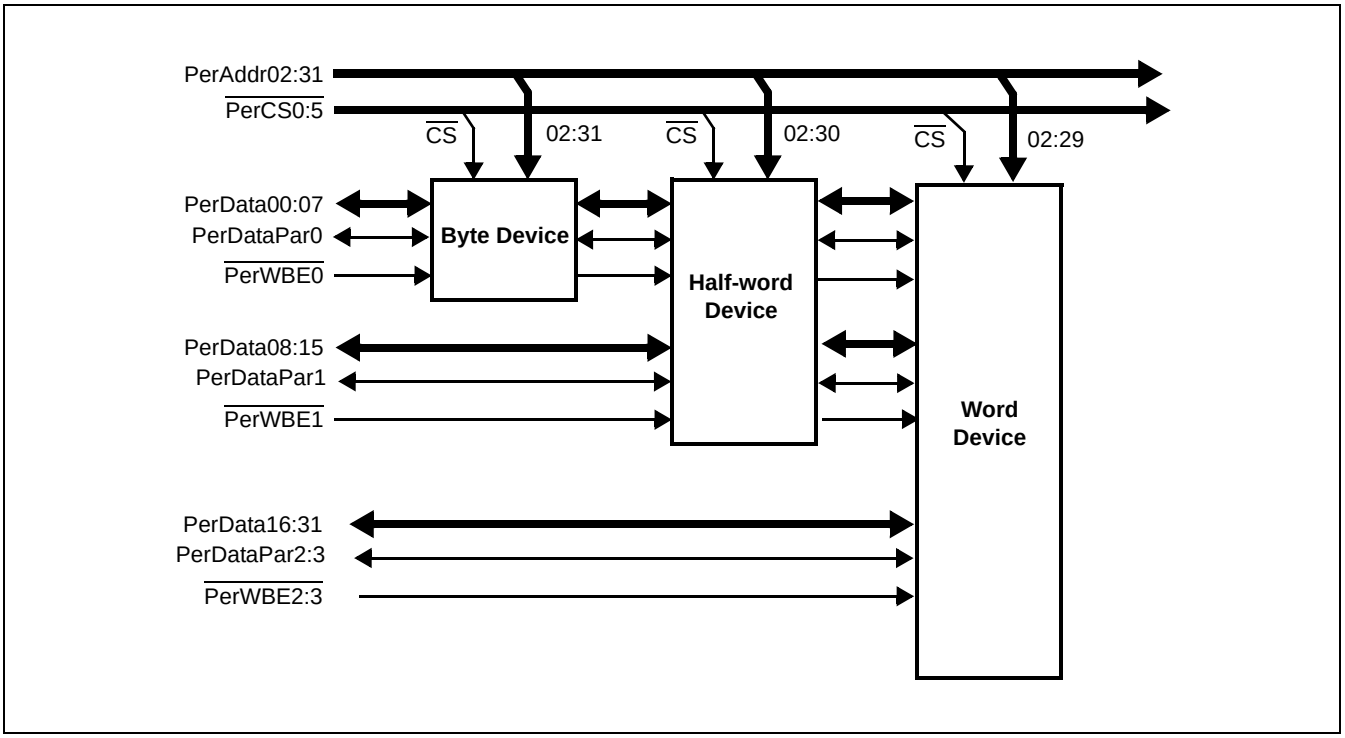
22.1.1 Interfacing to Byte, Half-Word, and Word Devices

Figure 22-2 illustrates how to interface byte, half-word, and word devices to the peripheral data bus. When devices are connected in this way, the EBC supports burst transfers and automatically converts read and write operations to the data width of the external device. As shown in *Figure 22-2*, half word devices should not connect to PerAddr31 and and word devices should not connect to PerAddr30:31. Instead, the active byte lanes should be inferred from PerWBE0:3, the read/write byte enables.

When a large number of byte-wide devices are attached to the peripheral data bus, the capacitive loading on byte lane 0 is much larger than the loading on byte lane 1, possibly resulting in unacceptable timing on byte lane 0.

If a bank register is configured as word-wide, then byte-wide devices may be attached to the bus in any byte lane (and accessed using byte loads and stores). External logic may be required to develop additional control signals if the data bus is used in this manner.

Figure 22-2. Attachment of Devices of Various Widths to the Peripheral Data Bus



22.1.2 Driver Enables

As shown in *Table 22-2* the output enables for the peripheral address, data and most of the EBC control signals are configurable. Setting EBC0_CFG[CSTC]=1 eliminates the need for pull-up resistors on PerCS0:5. Pull-ups are also unnecessary on the remainder of the EBC control signals when EBC0_CFG[EBTC]=1.

Both chip and system resets set EBC0_CFG[EBTC]=1 and EBC0_CFG[CSTC]=1. In most applications, clearing EBC0_CFG[CSTC] is not recommended. If EBC0_CFG[EBTC]=0, EBC control signals can change from the active state to high-Z without first being driven inactive. To prevent this, peripheral banks must be configured with at least one hold cycle (EBC0_BnAP[TH] > 0).

Table 22-2. Effect of Driver Enable Programming on EBC Signal States

EBC Operation	PerClk	PerCS0:5	PerAddr02:31 PerR/W PerWBE0:3 PerOE PerBLast	PerData0:31 PerDataPar0:3
Reset	Toggling	High-Z	High-Z	High-Z
Idle	Driven	EBC0_CFG[CSTC]	EBC0_CFG[EBTC]	EBC0_CFG[EBTC]
Read	Driven	Driven[1]	Driven[1]	High-Z
Write	Driven	Driven[1]	Driven[1]	Driven

1. Multiplexed EBC signals are not driven after reset and require pull-ups when used. Refer to the data sheet for pull-up recommendations.

If the EBC0_CFG bit is set, the signal is driven to the appropriate state during the indicated EBC operation. Otherwise, the I/O is High-Z.

22.2 Non-Burst Peripheral Bus Transactions

The timing of the PerCSn, PerOE, and PerWBE0:3 signals is programmable using the Peripheral Bank Access Parameter (EBC0_BnAP) registers. For non-burst transfers, the access parameter registers control the peripheral bus timing as follows:

- PerCSn becomes active 0–3 PerClk cycles (EBC0_BnAP[CSN]) after the address is driven.
- PerOE is driven low 0–3 PerClk cycles (EBC0_BnAP[OEN]) after PerCSn is active.
- PerBLast is active throughout the entire transfer and is driven high during the programmed hold time (EBC0_BnAP[TH]).
- PerWBE0:3 can be either write byte enables or read and write enables.

  If EBC0_BnAP[BEM] = 0, PerWBE0:3 are write byte enables and:

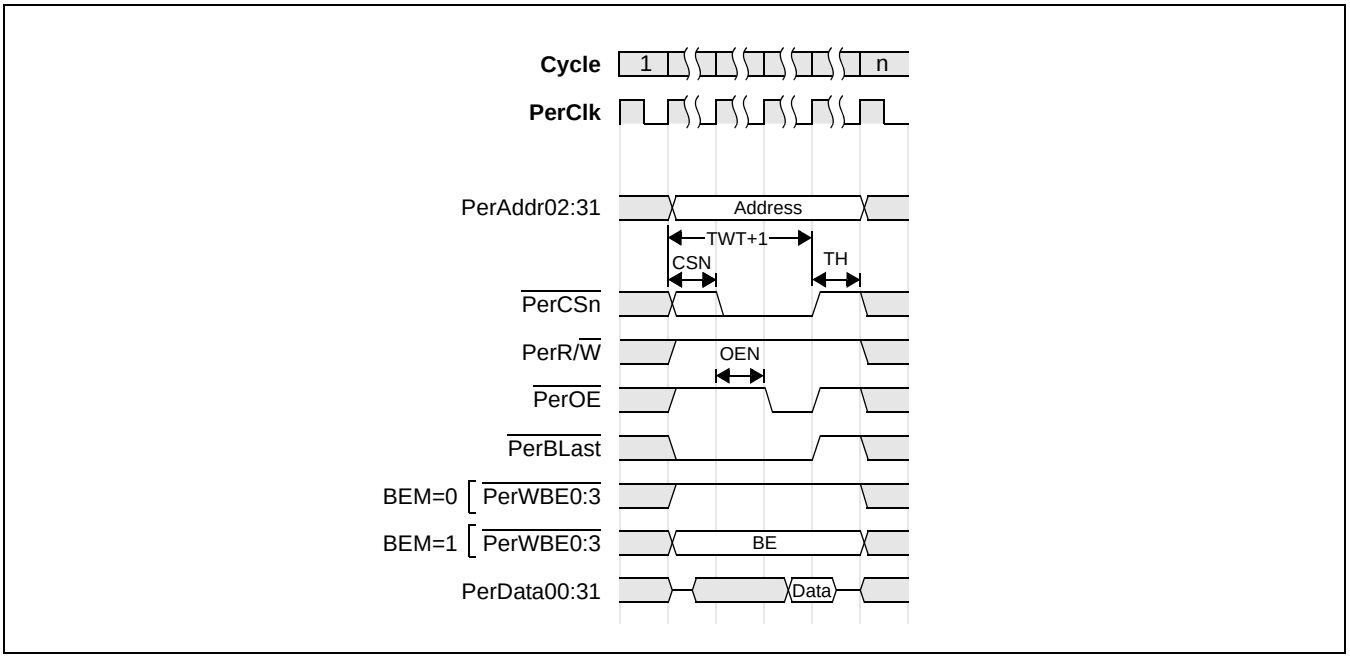
  - PerWBE0:3 goes active 0–3 (EBC0_BnAP[WBN]) PerClk cycles after PerCSn becomes active.
  - PerWBE0:3 becomes inactive 0–3 (EBC0_BnAP[WBF]) PerClk cycles before PerCSn becomes inactive.

  If EBC0_BnAP[BEM] = 1, PerWBE0:3 are read/write byte enables and have timing identical to the peripheral address bus. In this case the EBC0_BnAP[WBN] and EBC0_BnAP[WBF] parameters are ignored.
- 1–256 PerClk cycles (EBC0_BnAP[TWT] + 1) after the address became valid:
  - If EBC0_CFG[CSTC] = 1 or EBC0_BnAP[TH] > 0, PerCSn is driven high.
  - If EBC0_CFG[CSTC] = 0 and EBC0_BnAP[TH] = 0, PerCSn transitions directly from logic 0 to the high-impedance state.
- The parameters TWT, CSN, OEN, WBN, and WBF in EBC0_BnAP are not independent. For non-burst configured banks it is required that TWT ≥ CSN + MAX(OEN,WBN) + WBF.
- The hold time, EBC0_BnAP[TH], is programmable from 0–7 PerClk cycles. During the hold time, the peripheral address bus remains driven with the last address and all control signals are actively driven high. If the operation was a write, the peripheral data bus continues driving the last data value.
- There is no guarantee of dead cycles between transfers on the peripheral interface. If there are back-to-back transfers to the same memory bank and the number of hold cycles is programmed to zero (EBC0_BnAP[TH] = 0 and EBC0_BnAP[CSN] = 0), then:
  - PerCSn may not go inactive between the back-to-back transfers.
  - If EBC0_BnAP[OEN] = 0, PerOE may not become inactive between the two transfers.
  - If EBC0_BnAP[WBN] = 0 and EBC0_BnAP[WBF] = 0, PerWBE0:3 may not go inactive between the back-to-back transfers.

22.2.1 Single Read Transfer

Figure 22-3 shows the peripheral interface timing for a single read transfer from a non-burst enabled (EBC0_BnAP[BME]=0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[TWT]+1 cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input, PerErr. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles.

Figure 22-3. Single Read Transfer



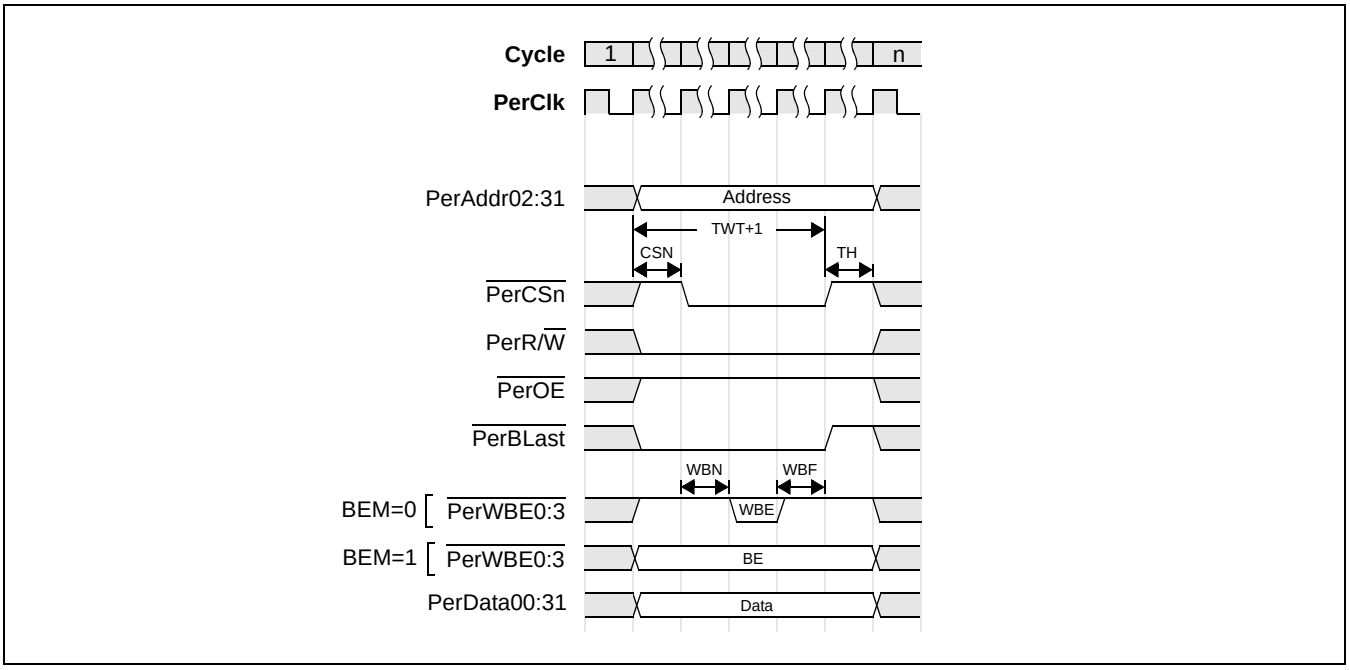
22.2.2 Single Write Transfer

Figure 22-4 shows the peripheral interface timing for a single write transfer to a non-burst enabled (EBC0_BnAP[BME]=0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

- If EBC0_BnAP[BEM]=0, byte enable mode is disabled and the PerWBE0:3 are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn. The EBC then waits until (EBC0_BnAP[TWT] – EBC0_BnAP[WBF] + 1) cycles have elapsed since the start of the transaction, then drives all the PerWBE0:3 inactive.
- If EBC0_BnAP[BEM]=1, the PerWBE0:3 lines are byte enables and have the same timing as the peripheral address bus.

After EBC0_BnAP[TWT+1] cycles elapse from the start of transfer, PerCSn and PerBLast are driven high. The EBC then waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 22-4. Single Write Transfer



22.3 Burst Transactions

Bursting is controlled on a per-bank basis by the Burst Mode Enable bit in the EBC0_BnAP registers. When enabled (EBC0_BnAP[BME]=1) this mode activates bursting for all cache line fills and flushes, PLB burst transfers to the EBC, and all packing and unpacking operations. When bursting is enabled:

- $\overline{\text{PerCSn}}$ becomes active 0–3 (EBC0_BnAP[CSN]) PerClk cycles after the address becomes valid.
- $\overline{\text{PerCSn}}$ is no longer actively driven:
  - 1–32 (EBC0_BnAP[FWT]+1) PerClk cycles after the address becomes valid when a single transfer occurs to a burst-enabled bank.
  - 1–8 (EBC0_BnAP[BWT]+1) PerClk cycles after the last address becomes valid during a burst:
    - If EBC0_CFG[CSTC]=1 or EBC0_BnAP[TH]>0, $\overline{\text{PerCSn}}$ is driven high.
    - If EBC0_CFG[CSTC]=0 and EBC0_BnAP[TH]=0, $\overline{\text{PerCSn}}$ transitions directly from logic 0 to the high-impedance state.
- During read operations $\overline{\text{PerOE}}$ is driven low 0–3 (EBC0_BnAP[OEN]) PerClk cycles after $\overline{\text{PerCSn}}$ is active. $\overline{\text{PerOE}}$ goes inactive when $\overline{\text{PerCSn}}$ goes inactive.
- For bursts, the EBC drives a new address (EBC0_BnAP[FWT]+1) + N*(EBC0_BnAP[BWT]+1) cycles after the start of the transaction, where N = 0, 1, 2, …
- Addresses during a burst may "wrap." For example, cache line fills are processed critical word first.
- During write operations, the write data is driven concurrent with each address.
- $\overline{\text{PerWBE0:3}}$ can be either write byte enables or read and write enables.

  If EBC0_BnAP[BEM]=0, $\overline{\text{PerWBE0:3}}$ are write byte enables and:

- For the first transfer of a burst, or a single transfer to a burst enabled bank, the appropriate write byte enables go low 0–3 (EBC0_BnAP[WBN]) cycles after PerCSn becomes active. The EBC then waits until EBC0_BnAP[FWT] – EBC0_BnAP[WBF] + 1 cycles have elapsed since the start of the transaction and drives PerWBE0:3 inactive.
- The remaining transfers of the burst are similar, except that PerWBE0:3 becomes active at the same time that each new address is driven on the interface. The PerWBE0:3 remain low for (EBC0_BnAP[BWT] + 1) – EBC0_BnAP[WBF] cycles.

If EBC0_BnAP[BEM]=1, PerWBE0:3 are byte enables that have timing identical to the peripheral address bus. In this case the EBC0_BnAP[WBN] and EBC0_BnAP[WBF] parameters are ignored.

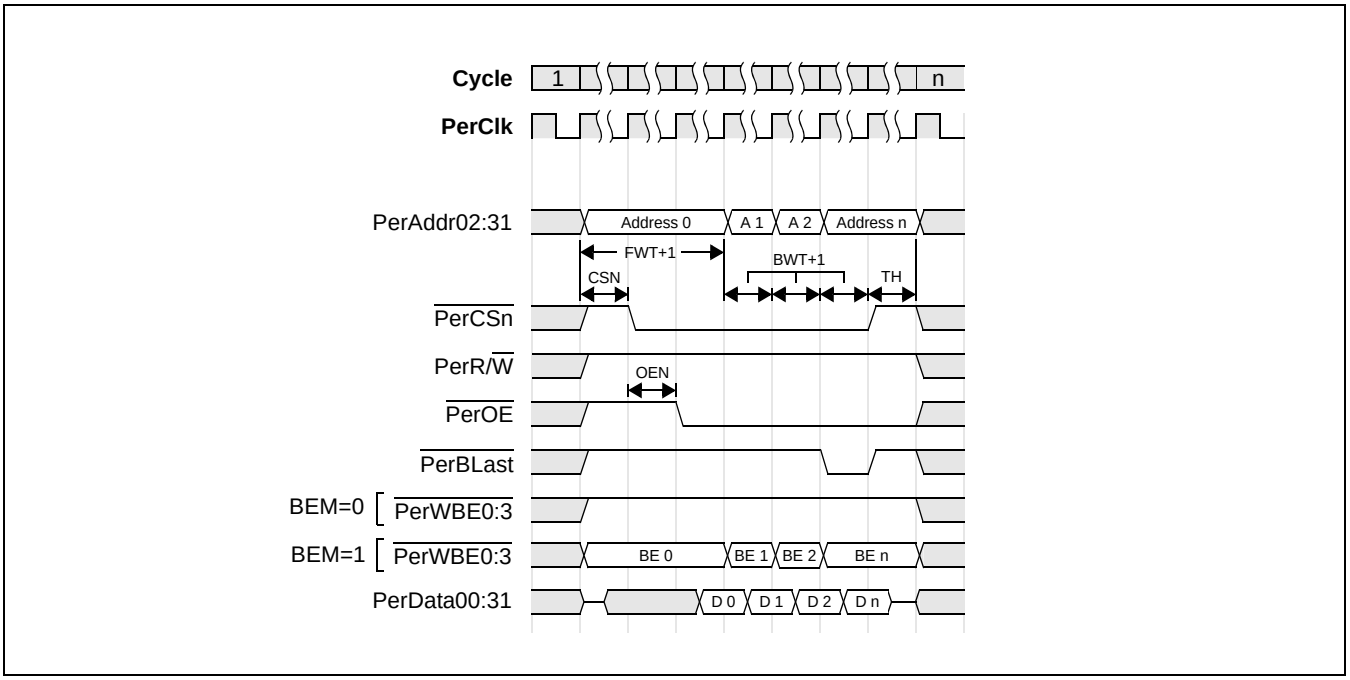
- PerBLast is active throughout the entire last (or only) transfer of a burst operation and is deactivated during the programmed hold time (EBC0_BnAP[TH]).
- Access bank parameters CSN, OEN and WBN apply to the first (or only) transfer of a burst, while WBF applies to all transfers. It is required that FWT ≥ CSN + MAX(OEN,WBN) + WBF and BWT ≥ WBF.
- Hold time (EBC0_BnAP[TH]) is programmable from 0 to 7 cycles. During the hold time, the peripheral address bus remains driven and all control signals are driven inactive. If the operation was a write, the peripheral data bus continues driving the write data.
- There is no guarantee of dead cycles between transfers on the peripheral interface. If there are back-to-back transfers to the same memory bank and the number of hold cycles is programmed to zero (EBC0_BnAP[TH]=0) and EBC0_BnAP[CSN]=0, then:
  - PerCSn may not go inactive between the back-to-back transfers.
  - If EBC0_BnAP[OEN]=0, PerOE may not become inactive between the two transfers.
  - If EBC0_BnAP[WBN]=0 and EBC0_BnAP[WBF]=0, PerWBE0:3 may not go inactive between the back-to-back transfers.

22.3.1 Burst Read Transfer

Figure 22-5 shows the peripheral interface timing for a burst read transfer from a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[FWT]+1 cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input, PerErr.

The next address of the burst is then driven and after EBC0_BnAP[BWT]+1 cycles the EBC performs the next read. The remaining items in the burst are read in the same manner, except that PerBLast is active during the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 22-5. Burst Read Transfer



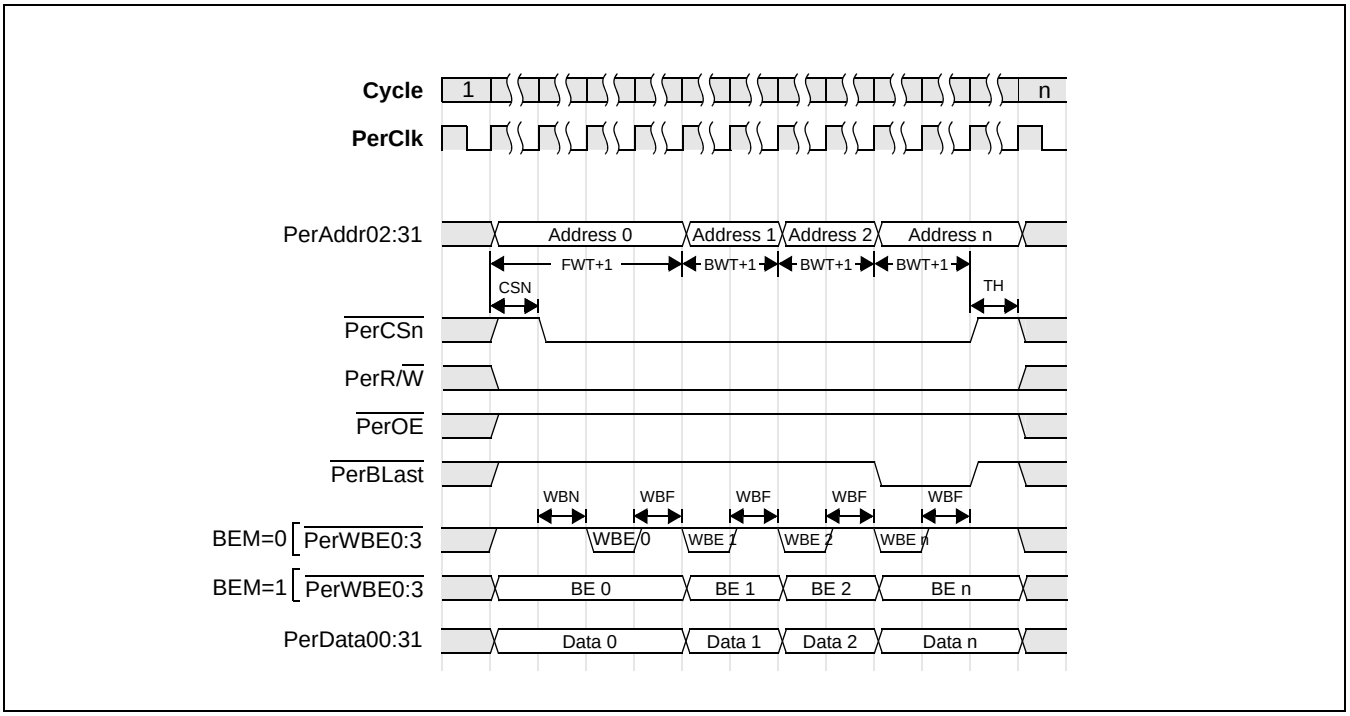
22.3.2 Burst Write Transfer

Figure 22-6 shows the peripheral interface timing for a burst write transfer to burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. At this point the signalling sequence depends on whether byte enable mode is enabled for the bank.

- If EBC0_BnAP[BEM]=0, byte enable mode is disabled and PerWBE0:3 are write byte enables. In this case, the appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn. The EBC then waits until (EBC0_BnAP[FWT] +1) - EBC0_BnAP[WBF] cycles have elapsed since the start of the transaction and drives PerWBE0:3 inactive. EBC0_BnAP[WBF] cycles are then allowed to elapse after which the address and data are output for the second element in the burst. As shown in *Figure 22-6*, the EBC transfers the subsequent data items in a similar manner.
- If EBC0_BnAP[BEM]=1, the PerWBE0:3 lines are byte enables and have the same timing as the peripheral address bus. In this configuration external logic may be necessary to latch write data at the appropriate times.

PerBLast goes low at the beginning of the last transfer to indicate that the burst is ending. The EBC then drives PerCSn and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 22-6. Burst Write Transfer



22.4 Device-Paced Transfers

For device-paced transfers, the EBC provides two distinct modes: Sample On Ready enabled and Sample On Ready disabled. The selection of these modes is controlled, for each bank, by EBC0_BnAP[SOR]. When Sample On Ready is enabled (EBC0_BnAP[SOR] = 1) data is transferred on the PerClk rising edge when PerReady is sampled active. When Sampling On Ready is disabled
(EBC0_BnAP[SOR] = 0), PerReady sampled active causes the data transfer to occur in the next cycle, which results in an additional cycle of wait time.

The ready signal (PerReady) is an input which allows the insertion of externally generated (device-paced) wait states. PerReady is monitored only when EBC0_BnAP[RE]=1.

- For burst disabled banks (EBC0_BnAP[BME] = 0) sampling of the PerReady input starts EBC0_BnAP[TWT] cycles after the beginning of the transfer. Wait states are inserted and sampling continues once per cycle until either PerReady is high when sampled or a timeout occurs.
- For burst enabled banks (EBC0_BnAP[BME] = 1) sampling of the PerReady input starts EBC0_BnAP[FWT] PerClk cycles after the beginning of the first transfer of a burst, and EBC0_BnAP[BWT] cycles after the beginning of subsequent transfers of the burst. Sampling continues once per cycle until either PerReady is sampled high or a timeout occurs.
- When EBC0_BnAP[SOR] = 1 data transfer occurs in the same cycle where PerReady is sampled active. In contrast, if EBC0_BnAP[SOR]=0 the data transfer occurs in the next cycle.
- When EBC0_BnAP[SOR] = 1, if the hold time is set to zero, EBC0_BnAP[TH] = 0, the programmed hold time is ignored and the EBC performs the transaction with one hold cycle.
- When EBC0_BnAP[RE] = 1, the Write Byte Enable Off parameter must be programmed to 0, EBC0_BnAP[WBF] = 0. As a result, during device-paced burst write transfers $\overline{\text{PerWBE0:3}}$ does not become inactive between data elements.

The EBC may be programmed to wait only a limited time for PerReady to become active, or it may be programmed for unlimited wait. If EBC0_CFG[PTD] = 1, timeouts are disabled and the EBC waits indefinitely for an active PerReady.

If EBC0_CFG[PTD] = 0, device-paced timeouts are enabled and the EBC only waits for the number of PerClk cycles programmed in EBC0_CFG[RTC]. The timeout counter is reset whenever the peripheral address changes. In this manner each data element within a device-paced burst transaction is treated separately for the purposes of determining whether a timeout error occurs. If PerReady does not become active before the timeout counter reaches the value programmed into EBC0_CFG[RTC], the transfer is aborted and an error is signalled. See *Error Reporting* on page 580 for details about how timeout errors are logged.

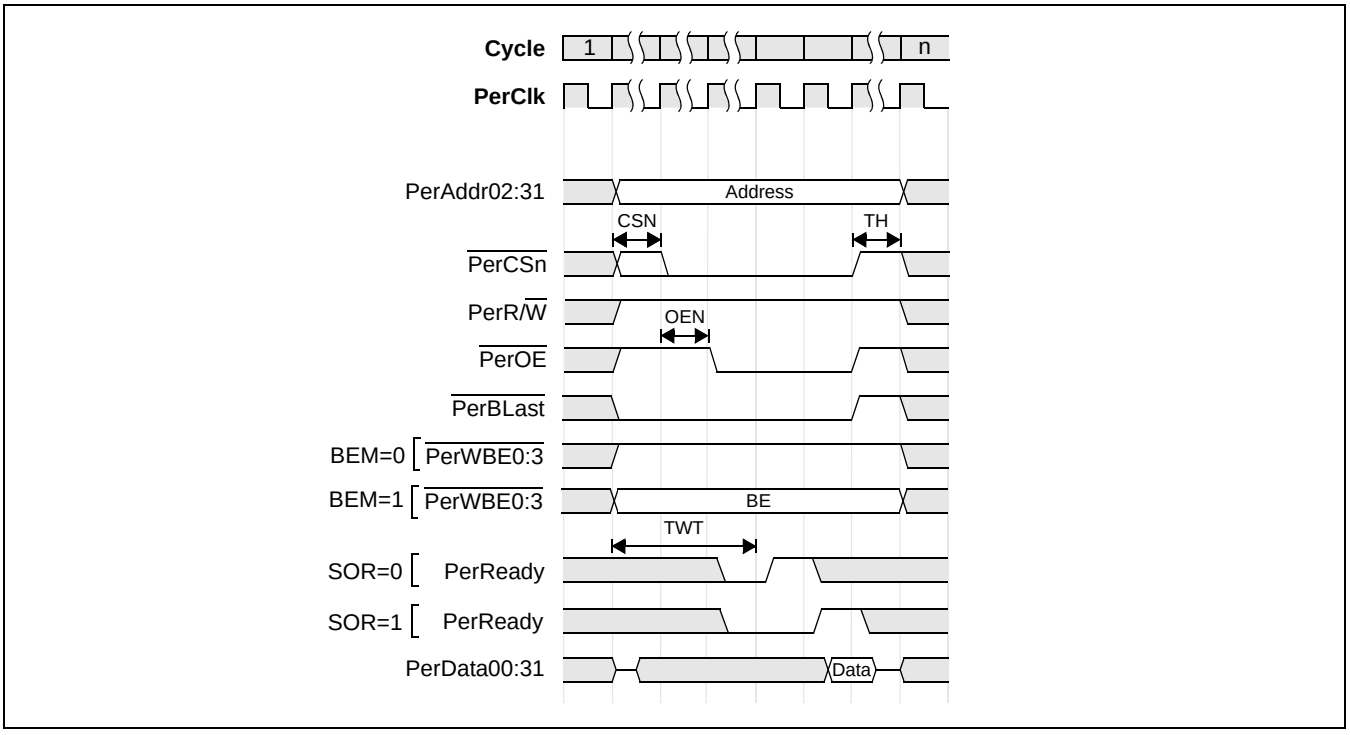
22.4.1 Device-Paced Single Read Transfer

Figure 22-7 shows the peripheral interface timing for a device-paced single read transfer from a burst disabled (EBC0_BnAP[BME]=0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

Once PerReady is sampled active if Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle. The EBC then samples the data bus and the peripheral error input, PerErr. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending EBC transfers to occur.

Figure 22-7. Device-Paced Single Read Transfer



22.4.2 Device-Paced Single Write Transfer

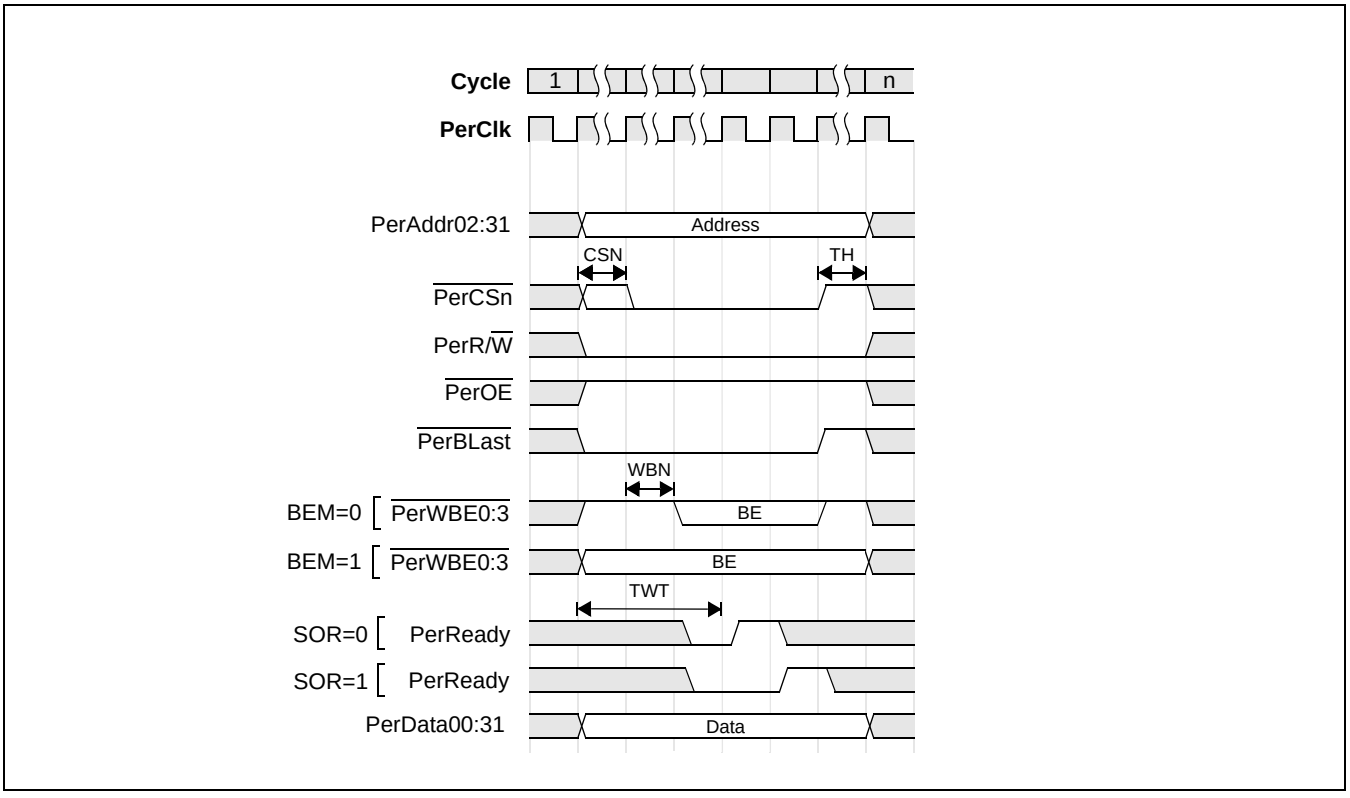
Figure 22-8 shows the peripheral interface timing for a device-paced single write transfer from a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. At this point the signalling sequence depends on whether byte enable mode is enabled for the particular bank.

- If EBC0_BnAP[BEM]=0, byte enable mode is disabled and the PerWBE0:3 are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn went low. PerWBE0:3 return high on the same PerClk edge that the write data is transferred (see below).
- If EBC0_BnAP[BEM]=1, the PerWBE0:3 lines are byte enables and have the same timing as the peripheral address bus.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sample PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle. At this point, the write transfer occurs and the EBC reads the peripheral error input, PerErr. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles.

Figure 22-8. Device-Paced Single Write Transfer



22.4.3 Device-Paced Burst Read Transfer

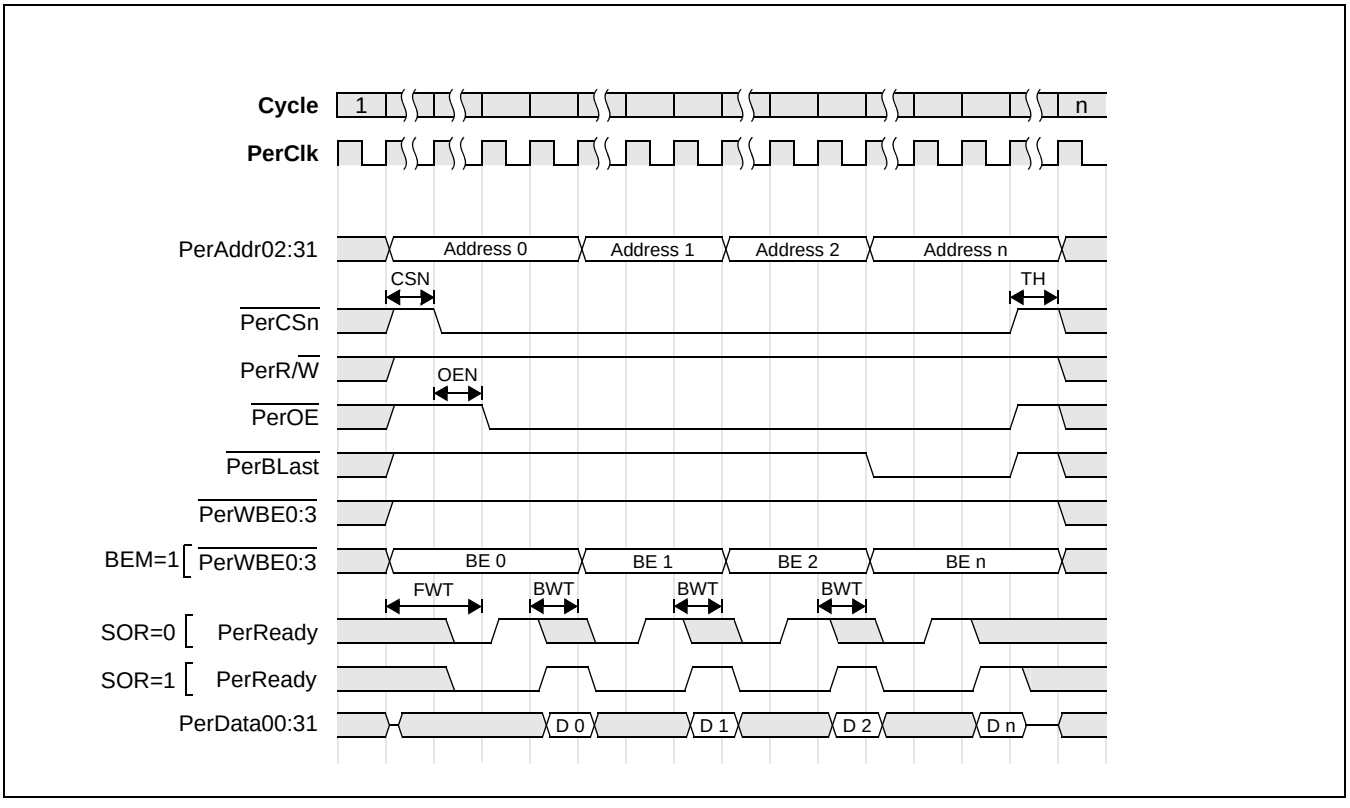
Figure 22-9 shows the peripheral interface timing for a device-paced burst read transfer from a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM]=1) the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn. The EBC then waits until EBC0_BnAP[FWT] cycles have elapsed since the start of the transaction and begins sampling PerReady.

If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle before sampling read data. The EBC then reads the data bus and the peripheral error input, PerErr.

The next address of the burst is then driven and after EBC0_BnAP[BWT] cycles the EBC waits for PerReady as described above. The remaining items in the burst are read in this same manner, except that PerBLast is active during the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 22-9. Device-Paced Burst Read Transfer



22.4.4 Device-Paced Burst Write Transfer

Figure 22-10 shows the peripheral interface timing for a device-paced burst write transfer to a burst enabled (EBC0_BnAP[BME]=1) bank. The transaction begins with the address being driven. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

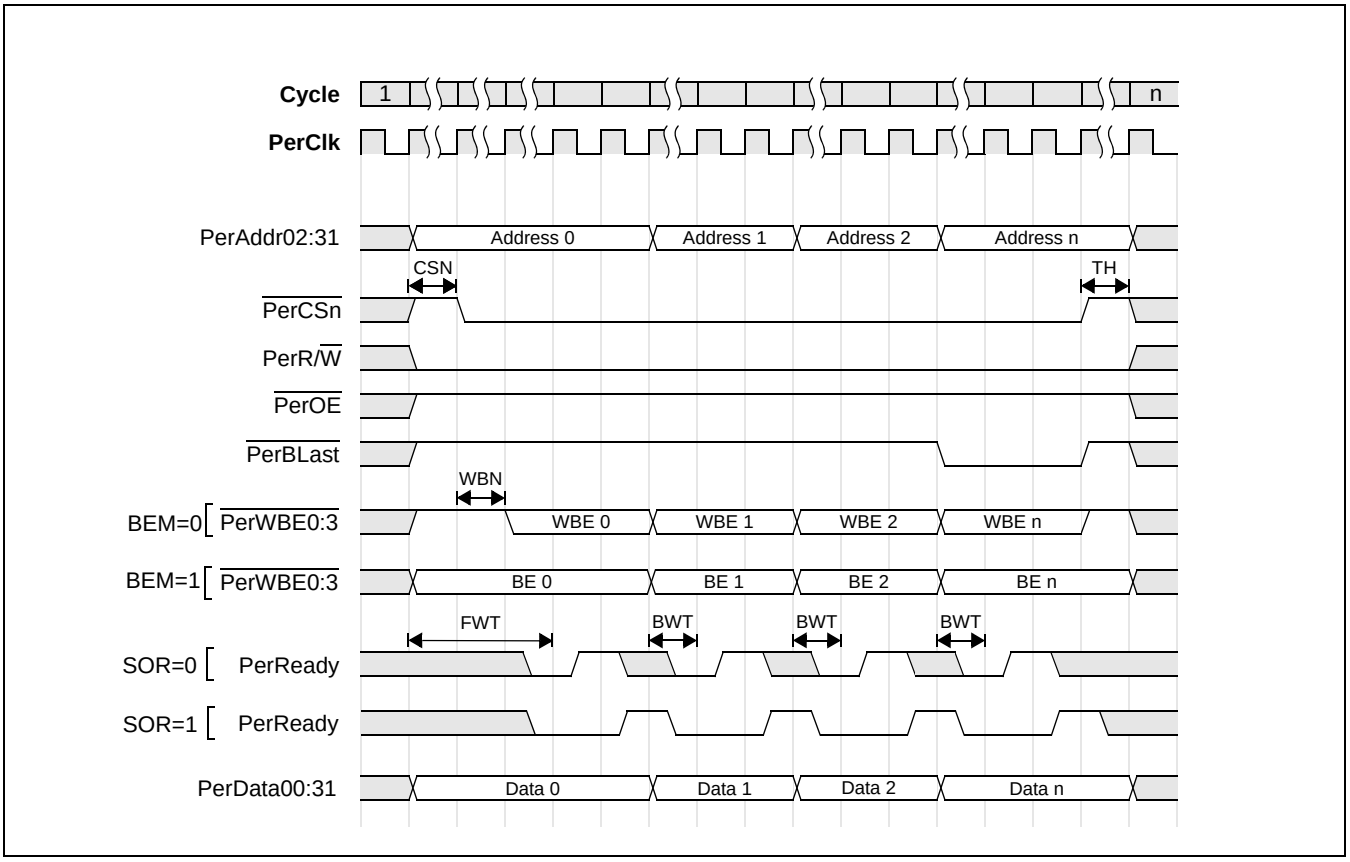
- If byte enable mode is disabled (EBC0_BnAP[BEM]=0) PerWBE0:3 are write byte enables. In this case the appropriate write byte enables go low EBC0_BnAP[WBN] cycles after PerCSn becomes active for the first element in a burst and EBC0_BnAP[WBN] cycles after each new address for the remainder of the burst. If EBC0_BnAP[WBN]<>0, PerWBE0:3 is driven inactive on the same PerClk edge that write data is transferred (see below). Otherwise, PerWBE0:3 remains low for all data elements in the burst.
- If EBC0_BnAP[BEM]=1, PerWBE0:3 are byte enables and have the same timing as PerAddr:31.

The EBC then waits until EBC0_BnAP[FWT] cycles have elapsed since the start of the transaction and begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD]=1) the EBC waits indefinitely for PerReady. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled (EBC0_BnAP[SOR]=0) the EBC waits one more cycle. At this point the write transfer occurs and the EBC reads the peripheral error input, PerErr.

The next address of the burst is then driven and after EBC0_BnAP[BWT] cycles the EBC waits for PerReady as described above. The remaining items in the burst are transferred in this same manner, except that PerBLast is active for the last data element. The EBC then drives PerCSn, PerOE and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 22-10. Device-Paced Burst Write Transfer

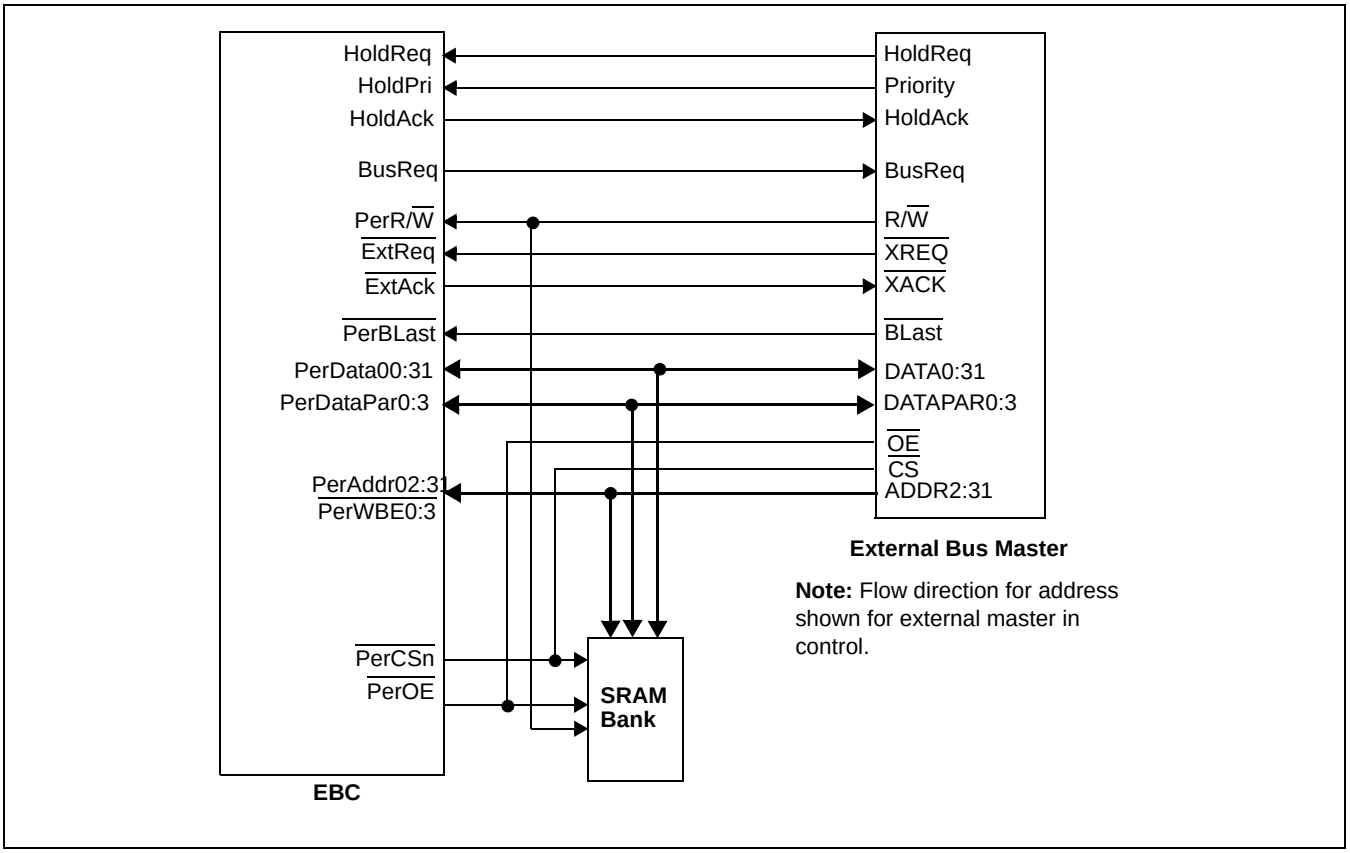


22.5 External Bus Master Interface

The EBC includes an External Bus Master (EBM) interface supporting a shared bus protocol which allows an EBM to gain control of the peripheral bus. Once an external master has been granted access to the peripheral interface it can read and write all PLB- and OPB-addressable memory, with the exception of devices controlled by the EBC. SDRAM memory is the only destination for an EBM transaction. For EBC-attached peripherals and memory, the external master is required to directly control the target.

Figure 22-11 shows a sample interconnection of the EBC, one SRAM bank, and one external bus master. While only one SRAM bank is shown, the bus master could access all eight of the SRAM banks. Also, with the appropriate arbitration logic, multiple bus masters may be used in a system.

Figure 22-11. Sample External Bus Master System



The EBM interface includes both arbitration and data transfer functions. The arbiter grants the peripheral bus to either the EBC or an external master, while the data path logic implements an external master to PLB bridge function. This bridge function translates the EBM interface protocol to that required by the on-chip PLB bus.

22.5.1 Arbitration

To gain control of the peripheral bus, the external bus master places an active level on the HoldReq input and drives the HoldPri signal to the desired value (see *Figure 22-12*). The HoldPri input selects the priority of the external master's transactions relative to other EBC operations. Internal to the PPC440EPx/GRx all reads and writes that target the EBC are assigned the priority 0b10.

If HoldPri=0 the external master priority is set to EBC0_CFG[EMPL], whereas HoldPri=1 sets the priority to EBC0_CFG[EMPH]. *Table 22-3* details the only unique ways of programming the external master priority fields in EBC0_CFG, along with the arbiter's response.

Table 22-3. External Master Arbitration.

EMPL	EMPH	HoldPri	When HoldReq=1 HoldAck Becomes Active	When HoldAck=1 BusReq Goes Active
0b11	0b11	X	After any active EBC operation completes.	Never.
0b00	0b11	0	If no EBC transfers are active or pending.	If an EBC transfer is pending.
		1	After any active EBC operation completes.	Never.
0b00	0b00	X	If no EBC transfers are active or pending.	If an EBC transfer is pending.

When an external bus master requests the bus by driving HoldAck=1, the EBC finishes any transfer in progress (except for bursts) before arbitrating between any pending PLB request and the external master request. If the external master presents a higher priority request during a PLB burst, the burst is terminated. Note that processor cache reads and writes are PLB line operations, not PLB bursts, and are never interrupted.

As detailed in *Table 22-4*, the arbitration logic drives HoldAck active either when the current EBC transaction completes or when there are no EBC transfers pending. Once the external master has been granted the bus, it may either directly control devices on the peripheral bus or read and write PLB- and OPB-addressable memory. If the EBC detects a pending PLB request when the external master owns the peripheral bus (HoldAck=1), BusReq may be asserted to signal the external master that the PPC440EPx/GRx wants to regain ownership of the peripheral bus. Table 22-3 lists the cases where a PLB request causes BusReq to go active. To ensure fairness and optimize bus utilization, an external master that receives an active BusReq should complete its current transaction and then relinquish ownership of the peripheral bus by driving HoldReq inactive. This is important since the EBC cannot reclaim ownership of the external bus until the external master negates HoldReq.

Table 22-4 details the usage of the EBC I/Os when an external master has been granted ownership of the peripheral bus. The usage statements in this table for signals other than HoldPri, HoldReq, HoldAck and BusReq do not apply if an external master directly controls a device on the peripheral bus.

Table 22-4. Signal States During Hold Acknowledge (HoldAck=1)

Signal Name	State	Usage
HoldPri	Input	Requested priority for external master tenure. HoldPri must not change state when HoldReq=1.
HoldReq	Input (=1)	External master must maintain its request when it owns the bus.
HoldAck	Output (=1)	External master is receiving an active bus grant.
BusReq	Output	See *Table 22-3*
ExtReq	Input	Indicates the external master is ready to transfer data.
ExtAck	Output	Indicates to the external master that a data transfer occurred.
PerAddr02:31	Input	Requested address from external master.
$\overline{\text{PerWBE0:3}}$	Input	Selects the requested byte(s) for reads and writes.
PerR/$\overline{\text{W}}$	Input	Determines if the operation is a read or write.
$\overline{\text{PerBLast}}$	Input	Active during single transfers and the last transfer of a burst.
PerData00:31	I/O	Read and write data.
PerDataPar0:3	I/O	Read and write data parity.
PerOE	High-Z	Unused.
PerErr	High-Z	Unused.

22.5.2 Transaction Overview

The EBM interface supports direct attachment of 8-, 16-, and 32-bit masters. By programming the width of the external master into EBC0_CFG[EMS] the interface accepts write data and provides read data at the appropriate width for the master. The EBM interface includes a 32-byte data buffer, used for both read and write operations between the EBM and PLB- and OPB-mapped memory locations. While write operations only use the buffer during bursts, all reads prefetch one double word and burst reads prefetch EBC0_CFG[BPF] double words from the source memory into the buffer. This prefetched data remains in the buffer until either a write operation is performed or a read is requested to a different 32-byte block of memory.

To provide the best possible performance, the EBM interface supports both single and burst transactions. Single read transfers result in the EBM reading and buffering a 64-bit double word from the requested memory address. The requested read data is then serviced from within this double word. If the next operation on the EBM interface is a read and targets this same double word, it is serviced directly from the buffer. Burst reads are similar, except that the EBM prefetches four double words beginning with the requested word.

Single write transfers result in a separate PLB transaction for each data item. To improve performance, burst writes are gathered in the 32-byte buffer and forwarded in a single PLB transaction to the target memory.

22.5.3 Single Read and Single Write Transfers

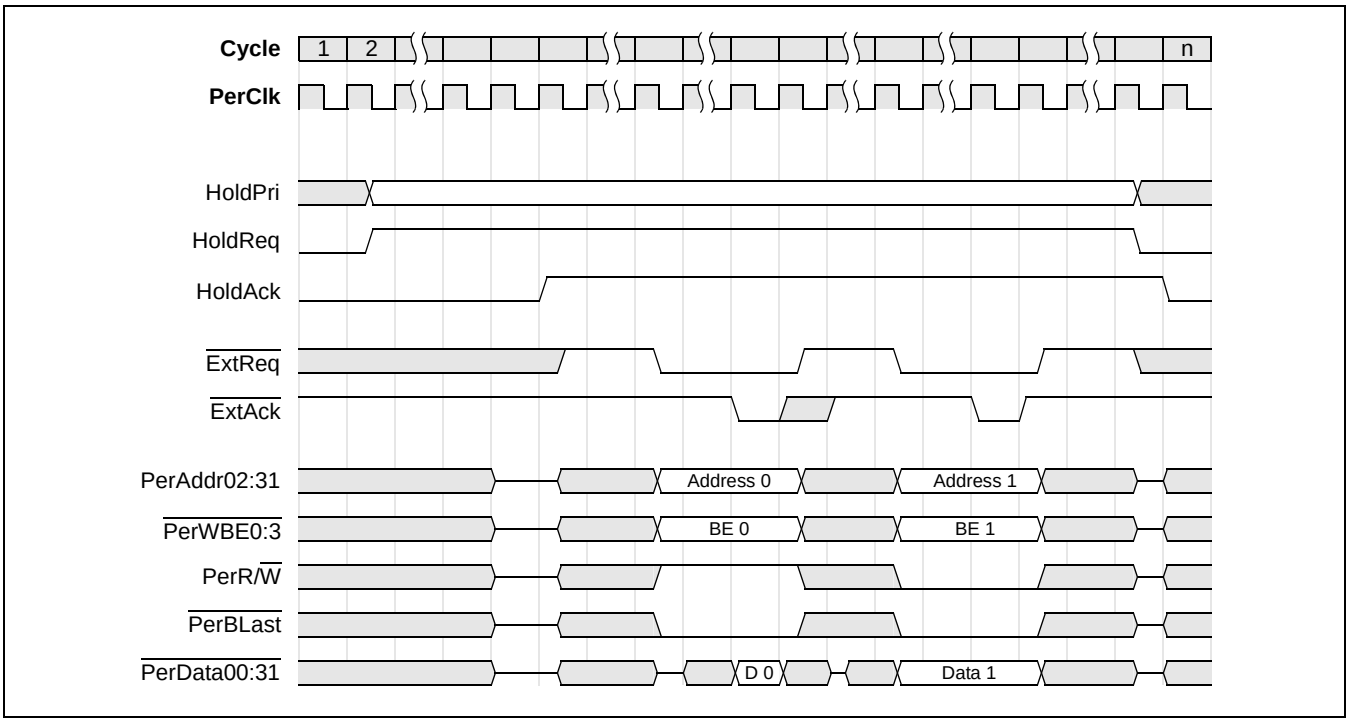
Figure 22-12 illustrates external master bus arbitration along with a single read and signal write transfer. An external master requests ownership of the peripheral bus by driving HoldReq active along with the desired priority on HoldPri. Observe that HoldPri must be held at constant value throughout the entire external master tenure. After two or more PerClk cycles the arbiter will grant the peripheral bus to the external master. The delay from when the external master asserts HoldReq to when HoldAck becomes active is variable and depends on any EBC transaction that may be in progress or pending, the level on HoldPri, and the programming in EBC0_CFG[EMPL] and EBC0_CFG[EMPH].

Once the external master is granted the peripheral bus (HoldAck=1), it may either directly control a device on the EBC or issue read and write transactions to the external master interface. This waveform and the ones that follow apply only to the later case. Additionally, cycles shown with breaks in the clock may not be present or may extend for multiple cycles.

To perform a single read operation the external master must:

- Place the desired address on PerAddr02:31.
- Indicate the requested data byte(s) on PerWBE0:3.
- Drive PerR/W high.
- Assert PerBLast to mark this as a single transfer.
- Request the transfer by driving and holding ExtReq low.

Figure 22-12. External Master Arbitration, Single Read and Single Write



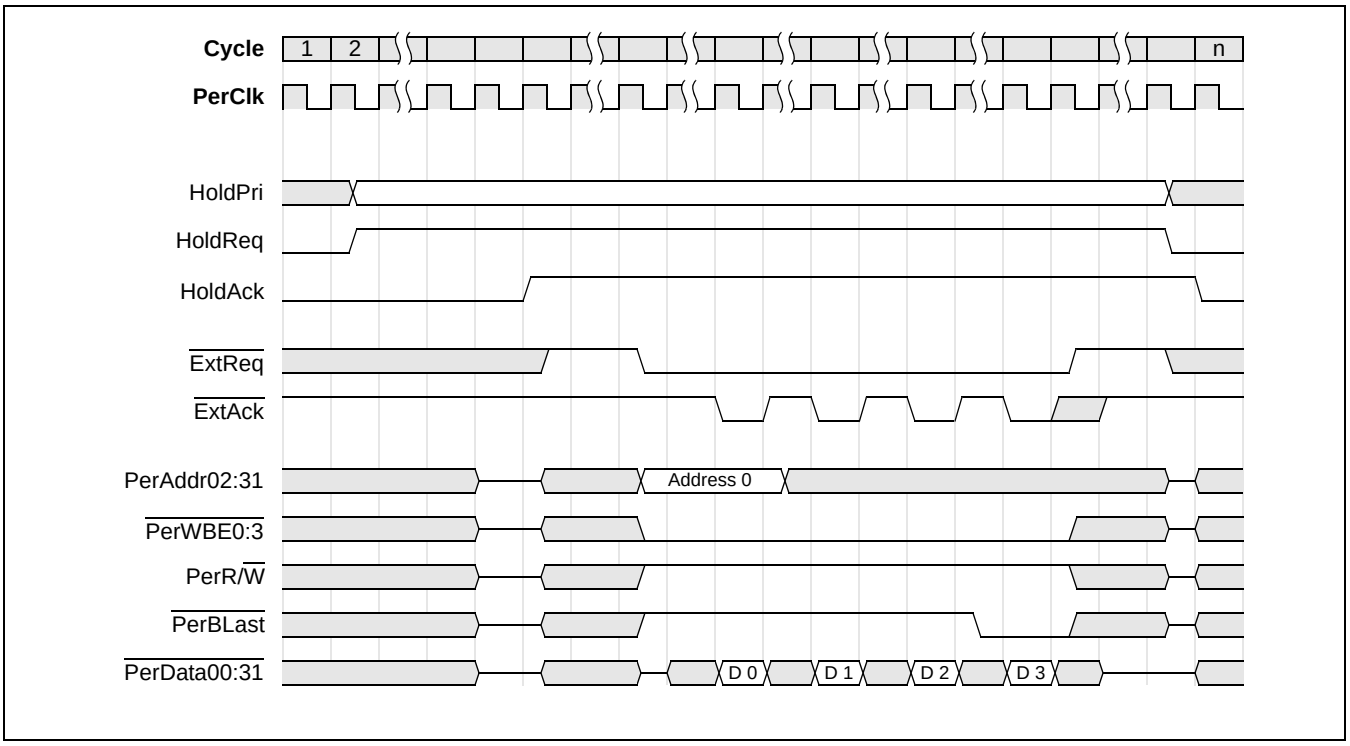
The EBM interface then converts this read request into a PLB transaction and reads the target memory location. ExtAck then goes low when read data is available on PerData. Note that the external master must not remove ExtReq until the cycle after ExtAck becomes active. In addition, ExtReq must be high for at least one cycle between all external master transactions.

Write transfers are similar to reads except that the write data must be provided along with the address and PerR/$\overline{W}$ is low to indicate a write. As with reads, ExtReq must be held until the cycle after ExtAck is received.

22.5.4 Burst Read Transfer

Burst reads are preferred when accessing sequential addresses as they provide much better performance. *Figure 22-13* illustrates an external master burst read transaction. A burst read differs from a single read in that PerBLast is held inactive for all but the last transfer of the burst. In addition, the EBM only requires the address of the first burst element. Following each ExtAck the EBM uses the size of the master configured in EBC0_CFG[EMS] to increment its internal address counter as appropriate.

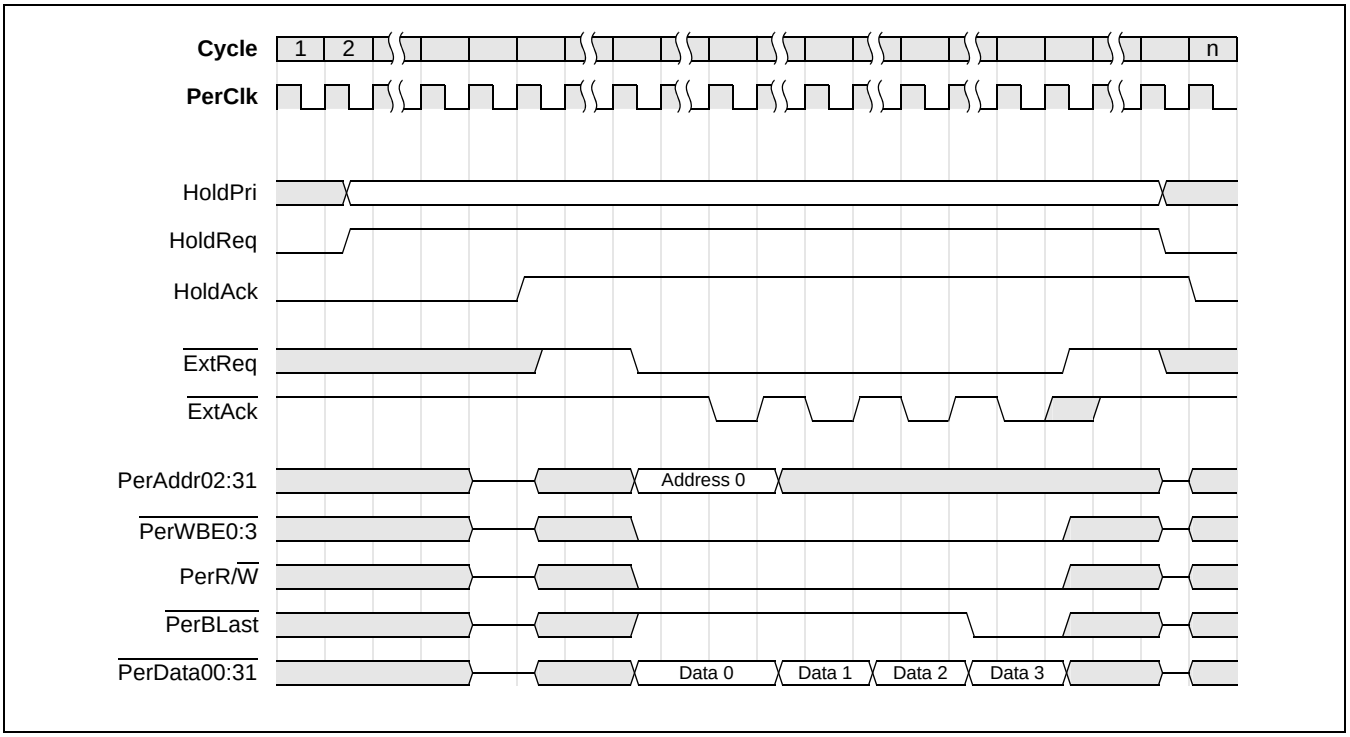
Figure 22-13. External Master Burst Read



22.5.5 Burst Write Transfer

Burst writes are preferred when accessing sequential addresses as they provide much better performance. *Figure 22-14* illustrates an external master burst write transaction. A burst write differs from a single write in that PerBLast is held inactive for all but the last transfer of the burst. In addition, the EBM only requires the address of the first burst element. Following each ExtAck the EBM uses the size of the master in EBC0_CFG[EMS] to increment its internal address counter as appropriate.

Figure 22-14. External Master Burst Write



22.5.6 External Master Error Interrupts

The EBM can generate an interrupt if a PLB error is encountered while read or writing data. As example, an DDR SDRAM uncorrectable ECC error will cause and EBM interrupt. Please refer to the UIC Chapter for additional information regarding interrupts.

22.6 EBC Registers

All EBC configuration registers are accessed with the move-to-DCR (**mtdcr**) and move-from-DCR (**mfdcr**) instructions using indirect addressing. In indirect addressing, the **mtdcr** and **mfdcr** instructions access a given EBC0 register via its address offset which is contained in the EBC0_CFGADDR register. The data for the specified register is moved to or moved from the EBC0_CFGDATA register as described below.

Table 22-5. EBC DCR Addresses

Register	DCR Address	Access	Description
EBC0_CFGADDR	0x0012	R/W	External Bus Controller Configuration Address Register
EBC0_CFGDATA	0x0013	R/W	External Bus Controller Configuration Data Register

Figure 22-15. EBC Configuration Address Register (EBC0_CFGADDR)			
0:16		Reserved	
17:31	DCRA	DCR Address Offset	

Figure 22-16. EBC Configuration Data Register (EBC0_CFGDATA)			
0:31	DCRD	DCR Data Port	

Table 22-6 lists the indirectly accessed EBC configuration and status registers.

Table 22-6. EBC Configuration and Status Registers

Mnemonic	Address Offset	Access	Description	Page
EBC0_B0CR	0x0000	R/W	Peripheral Bank Configuration Register 0	577
EBC0_B1CR	0x0001	R/W	Peripheral Bank Configuration Register 1	577
EBC0_B2CR	0x0002	R/W	Peripheral Bank Configuration Register 2	577
EBC0_B3CR	0x0003	R/W	Peripheral Bank Configuration Register 3	577
EBC0_B4CR	0x0004	R/W	Peripheral Bank Configuration Register 4	577
EBC0_B5CR	0x0005	R/W	Peripheral Bank Configuration Register 5	577
EBC0_B0AP	0x0010	R/W	Peripheral Bank Access Parameter 0	578
EBC0_B1AP	0x0011	R/W	Peripheral Bank Access Parameter 1	578
EBC0_B2AP	0x0012	R/W	Peripheral Bank Access Parameter 2	578
EBC0_B3AP	0x0013	R/W	Peripheral Bank Access Parameter 3	578
EBC0_B4AP	0x0014	R/W	Peripheral Bank Access Parameter 4	578
EBC0_B5AP	0x0015	R/W	Peripheral Bank Access Parameter 5	578
EBC0_BEAR	0x0020	R	Peripheral Bus Error Address Register	580
EBC0_BESR0	0x0021	R	Peripheral Bus Error Status Register 0	580
EBC0_CFG	0x0023	R/W	EBC Configuration Register	576

To access an indirectly accessed register, software must first write its address offset into the EBC0_CFGADDR register. The target register can then be read or written through the EBC0_CFGDATA DCR address. The following PowerPC code illustrates this procedure by writing the EBC0_B0CR register and then reading back the written value.

```
li        r3,EBC0_B0CR              ! address offset
lis       r4,<config upper>         ! upper 16-bits of configuration data
ori       r4,r4,<config lower>      ! lower 16-bits of configuration data
mtdcr     EBC0_CFGADDR,r3           ! set offset addr
mtdcr     EBC0_CFGDATA,r4           ! write config data
mfdcr     r5,EBC0_CFGDATA           ! read back config data
```

22.6.1 EBC Configuration Register (EBC0_CFG)

The contents of EBC0_CFG are accessed indirectly, using the EBC0_CFGADDR and EBC0_CFGDATA.

Figure 22-17. EBC Configuration Register (EBC0_CFG)

0	EBTC	External Bus Three-State Control 0 Address, data and control signals are high-Z between EBC transfers. 1 Between EBC transfers the peripheral data bus, address bus and control signals are driven.	Default after reset is EBTC = 1. Set EBTC = 0 to allow an external bus master to drive the peripheral bus.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	If PTD = 1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	If PTD = 0, the number of cycles from PerAddr02:31 changing until a timeout error occurs.
5:6	EMPH	External Master Priority High Allowed values: 00 01 10 11	This two-bit field contains the PLB priority that will be driven onto the PLB for a cycle from an External Master accessing a PLB slave when the External Master Priority input signal is high.
7:8	EMPL	External Master Priority Low Allowed values: 00 01 10 11	This two-bit field contains the PLB priority that will be driven onto the PLB for a cycle from an External Master accessing a PLB slave when the External Master Priority input signal is low.
9	CSTC	Chip Select Three-state Control 0 PerCS0:5 are high-Z between EBC transfers. 1 PerCS0:5 are always driven.	Default after reset is CSTC = 1. Set CSTC = 0 to allow an external bus master to drive the peripheral bus.
10:11	BPF	Burst Prefetch 00 Prefetch 1 double word 01 Prefetch 2 double words 10 Prefetch 4 double words 11 Reserved	Controls the amount of data prefetching when the EBC is servicing a PLB burst read. For most applications set this field to 0b00.
12:13	EMS	External Master Size 00 8-bit external master 01 16-bit external master 10 32-bit external master 11 No external master	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0000–1111	The EBC makes a sleep request to the Clock and Power Management unit when PME = 1 and the EBC has been idle for 32 × PMT PerClk cycles.
20:31		Reserved	

22.6.2 Peripheral Bank Configuration Registers (EBC0_B0CR:EBC0_B5CR)

These registers must be configured to access memory in a bank. Boot ROM, if installed, must be attached to bank 0. If a boot ROM is installed, EBC0_B0CR[BAS] is loaded with a value of 0xFFE, and EBC0_B0CR[BS] is loaded with a value of 0b001 (2MB) immediately following SysReset inactive.

Reset Value for bank 0:

EBC0_B0CR = OxFFE28000 (8 bit boot source)

EBC0_B0CR = 0xFFE2A000 (16 bit boot source)

EBC0_B0CR = 0xFFE2C000 (32 bit boot source and Boot from NAND Flash)

Figure 22-18. Peripheral Bank Configuration Registers (EBC0_B0CR–EBC0_B5CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be aligned to the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs when a write is attempted to a read-only bank or a read from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 32-bit bus (used when NDFC is configured for this bank) 11 Reserved	The boot ROM must be attached to bank 0 as it is the only bank configured after a reset. Its bus width is controlled by strapping pins.
19:31		Reserved	

- **BAS (Base Address Select, bits 0:11)** – Sets the base address for a peripheral device. The bank starting address must be a multiple of the bank size programmed in the BS field. The BAS field is compared to bits 0:11 of the address. If the address is within the range of a BAS field, the associated bank is enabled for the transaction.

  Multiple bank registers may be inadvertently programmed with the same base address or as overlapping banks. An attempt to use such overlapping banks is recorded in EBC0_BESR0 as a configuration error and no external access occurs. This error may result in a machine check exception if the requesting master is the CPU. If the error occurred during a DMA access, the DMA may signal an interrupt to the PPC440EPx/GRx through the UIC.
- **BS (Bank Size, bits 12:14)** – Sets the number of bytes which the bank may access, beginning with the base address set in the BAS field.
- **BU (Bank Usage, bits 15:16)** – Protects banks of physical devices from read or write accesses.
- When a write access is attempted to an address within the range of the BAS field, and the bank is designated as read-only, a protection error occurs. Also, when a read access is attempted to an address within the range

of the BAS field, and the bank is designated as write-only, a protection error occurs. The address of the attempted access is logged in EBC0_BEAR and type of error is logged in EBC0_BESR0.

- **BW (Bus Width, bits 17:18**) – Controls the width of region accesses. If the BW field is 0b00, the region is configured for an 8-bit data bus; 0b01 indicates a 16-bit data bus; 0b10 indicates a 32-bit data bus width. If devices are attached to the data bus as shown in *Figure 22-2*, the EBC automatically packs read data and unpacks write data when a data transfer size mismatch exists.

22.6.3 Peripheral Bank Access Parameters (EBC0_B0AP:B5AP)

The following figure describes EBC0_BnAP register bits.

Reset value for EBC0_B0AP = 0x7E8FFE80

Figure 22-19. Peripheral Bank Access Parameters (EBC0_B0AP–EBC0_B5AP)

0	BME	Burst Mode Enable 0 Bursting is disabled 1 Bursting is enabled	
1:8	TWT	Transfer Wait 0–255 PerClk cycles	Wait states on all transfers when BME=0.
9:11		Reserved	
12:13	CSN	Chip Select On Timing 0–3 PerClk cycles	Number of cycles from peripheral address driven to PerCSn low.
14:15	OEN	Output Enable On Timing 0–3 PerClk cycles	Number of cycles from PerCSn low to PerOE low.
16:17	WBN	Write Byte Enable On Timing 0–3 PerClk cycles	If BEM=0, number of cycles from PerCSn low to PerWBE0:3 active.
18:19	WBF	Write Byte Enable Off Timing 0–3 PerClk cycles	If BEM=0 and RE=0, number of cycles PerWBEn becomes inactive prior to PerCSn inactive.
20:22	TH	Transfer Hold 0–7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer.
23	RE	Ready Enable 0 PerReady is disabled 1 PerReady is enabled	
24	SOR	Sample on Ready 0 Data transfer occurs one PerClk cycle after PerReady is sampled active 1 Data transfer occurs in the same PerClk cycle that PerReady becomes active	
25	BEM	Byte Enable Mode 0 PerWBE0:3 are only active for write cycles 1 PerWBE0:3 are active for read and write cycles	If BEM=0, PerWBE0:3 timing is controlled by WBN and WBF. If BEM=1, PerWBE0:3 has the same timing as PerAddr02:31.
26	PEN	Parity Enable 0 Disable Parity checking for this bus region 1 Enable Parity checking for this bus region	
27:31		Reserved	

- **BME (Burst Mode Enable, bit 0)** – Controls bursting for cache line fills and flushes, PLB burst transfers and all packing and unpacking operations. If BME=1, bursting is enabled. When bursting is enabled the parameters

Chip Select On (CSN), Output Enable On (OEN), and First Wait (FWT) apply only to the first transfer, while Burst Wait (BWT) and Write Byte Enable On (WBN) apply during all remaining transfers of the burst.

- **TWT (Transfer Wait, bits 1:8**) – Specifies the number of wait states taken by each transfer to the bank. The number of cycles from address valid to the deassertion of PerCSn is (1 + TWT), where $0 \leq TWT \leq 255$. This field is used for non-burst transfers (field BME = 0).
- **FWT (First Wait, bits 1:5)** – Specifies the number of wait states to be taken by the first access to the bank during a burst transfer (field BME = 1). During a burst the number of cycles from the first address valid to the second address is (1 + FWT), where $0 \leq FWT \leq 31$.
- **BWT (Burst Wait, bits 6:8)** – Specifies the number of wait states to be taken by accesses beyond the first during a burst transfer (field BME = 1). On burst accesses except for the last, the number of cycles from address valid to the next valid address on each burst access is (1 + BWT), where $0 \leq BWT \leq 7$. On the last burst access, the number of cycles from address valid to the deassertion of PerCSn is (1 + BWT), where $0 \leq BWT \leq 7$.
- **CSN (Chip Select On Timing, bits 12:13**) – Specifies the chip select turn on delay relative to the address. PerCSn may turn on coincident with the address or be delayed by 1, 2, or 3 PerClk cycles.
- **OEN (Output Enable On Timing, bits 14:15)** – Specifies when the output enable signal, PerOE, is asserted for read operations relative to the chip select signal. If 0, PerOE is asserted coincident with the chip select. If 1, 2 or 3, PerOE is delayed by 1, 2, or 3 PerClk cycles.
- **WBN (Write Byte Enable On Timing, bits 16:17**) – Specifies when the write byte enables, PerWBE0:, are asserted relative to the chip select signal. If 0, then PerWBE0: turns on coincident with the chip select. If 1, 2, or 3, PerWBE0: is delayed 1, 2, or 3 PerClk cycles from the chip select.
- **WBF (Write Byte Enable Off Timing, bits 18:19)** – Specifies when the write byte enables are deasserted, relative to the deassertion of the chip select signal. If WBF=0, PerWBE0: goes high coincident with the chip select signal. If WBF is 1, 2, or 3, PerWBE0: turns off 1, 2, or 3 PerClk cycles before the turn-off of the chip select signal.

  Note: It is an error to set WBF > BWT. Moreover, for device-paced transfers (EBC0_BnAP[RE]=1) WBF must be set to 0.
- **TH (Transfer Hold, bits 20:22)** – Specifies the number of PerClk cycles (0 through 7) that the peripheral bus is held idle after the deassertion of PerCSn. During these cycles, the address bus and data bus are active and PerR/W is valid. During the hold time, chip select, output enable, and write byte enables are inactive. If Ready Mode is used (RE=1) along with Sample on Ready (SOR=1) TH must be set to at least 1.
- **RE (Ready Enable, bit 23)** – Controls the use of the PerReady input signal. If RE=0, the PerReady input is ignored and no additional wait states are inserted into bus transactions. If RE=1, the PerReady input is examined after the wait period expires; additional wait states are inserted if the PerReady input is 0. The maximum number of wait states in each transaction is determined by the settings in the Device-Paced Timeout Disable (PTD) and Ready Timeout Counter (RTC) fields in EBC0_CFG. If EBC0_CFG[PTD] = 0, the PPC440EPx/GRx waits the number of cycles indicated by EBC0_CFG[RTC] for PerReady to become active. If EBC0_CFG[PTD] = 1, the ready timeout function is disabled and the PPC440EPx/GRx waits indefinitely until PerReady=1. If PerReady does not become active in the allotted time, the address of the error is logged in EBC0_BEAR and the type of error is captured in EBC0_BESR0.
- **SOR (Sample Ready, bit 24)** – Controls the location of the data transfer cycle with respect to the PerReady input. If SOR=1 the data transfer occurs on the same PerClk edge that PerReady is sampled active, whereas if SOR=0 the data transfer occurs one cycle later.
- **BEM (Byte Enable Mode, bit 25)** – Controls whether PerWBE0: is active during writes or for both reads and writes.
- **PEN (Parity Enable, bit 26)** – Enables the region for parity generation and checking.

22.7 Error Reporting

The EBC monitors three kinds of the following errors when performing read and write transfers. Of these four, bank protect and external bus errors are always checked, while timeout error checking must be enabled using DCR-mapped configuration registers.

- **Protect Error** – Requested read or write operation violates the bank usage programmed in EBC0_BnCR[BU]. For example, write attempt to read-only bank. In all cases, no external bus activity occurs.
- **Timeout Error** – This error is possible during memory operations when both PerReady sampling is enabled, EBC0_BnAP[RE]=1, and device paced timeouts are enabled, EBC0_CFG[PTD]=0. Whenever the peripheral address bus changes the EBC begins counting PerClk cycles. If the count reaches the value represented by EBC0_CFG[RTC] a timeout error occurs. Note that timeout errors are not possible during the peripheral portion of DMA transfers.
- **Parity Error** – When EBC0_BnAP[PEN] = 1.

  When the EBC slave detects one of the above errors it reports the error condition to the PLB master that initiated the transfer. The EBC also logs the type of error into EBC0_BESR0 and the address of the error in EBC0_BEAR.

22.7.1 Peripheral Bus Error Address Register (EBC0_BEAR)

The Peripheral Bus Error Address Register (EBC0_BEAR) is a 32-bit register containing the address of the access where a data bus error occurred. The contents of the EBC0_BEAR are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the mfdcr and mtdcr instructions.

The Bus Error Address Register (BEAR) is a 32-bit register which contains the address of the access where a data bus error occurred. The BEAR is written when a data access error occurs and its contents may be locked until the lock bit in the Bus Error Status Register 0 (BESR0) is cleared, depending on the state of the PLB_lockErr signal when the transaction was accepted. The contents of the BEAR can be accessed via the move from device control register (**mfdcr**) and move to device control register (**mtdcr**) instruction.

Figure 22-20. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error (asynchronous)	

22.7.2 Peripheral Bus Error Status Register 0 (EBC0_BESR0)

The Peripheral Bus Error Status Register 0 (EBC0_BESR0) records the occurrence and type of errors for transactions attempted on behalf of the PLB4 to PLB3 bridge, MAL, PCI bridge, DMA controller, and OPB to PLB3 bridge. The contents of EBC0_BESR0 are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

It is possible to have both a parity error and a bus error on a single transfer. If the bus error and parity error are both activated as a result of an access, then the bus error will be detected first and the BESR and the BEAR will be updated and locked with the bus error in the type field. Both the bus error and parity error will be reported, but the error type field will not indicate a parity error.

Figure 22-21. Peripheral Bus Error Status Register 0 (EBC0_BESR0)			
0:2	EET0	Error type for master 0 000 No error 001 Parity Error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 0 - PLB4 to PLB3 bridge
3	RWS0	Read/write status for master 0 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL0	Field lock for master 0 0 EET0 and RWS0 fields are unlocked 1 EET0 and RWS0 fields are locked	
5	AL0	EBC0_BEAR address lock for master 0 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
6:11		Reserved	
12:14	EET2	Error type for master 2 000 No error 001 Parity Error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 2 - PCI bridge
15	RWS2	Read/write status for master 2 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL2	Field lock for master 2 0 EET2 and RWS2 fields are unlocked 1 EET2 and RWS2 fields are locked	
17	AL2	EBC0_BEAR address lock for master 2 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
18:20	EET3	Error type for master 3 000 No error 001 Parity Error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 3 - DMA controller
21	RWS3	Read/write status for master 3 0 Error operation was a write operation 1 Error operation was a read operation	
22	FL3	Field lock for master 3 0 EET3 and RWS3 fields are unlocked 1 EET3 and RWS3 fields are locked	

23	AL3	EBC0_BEAR address lock for master 3 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
24:31		Reserved	

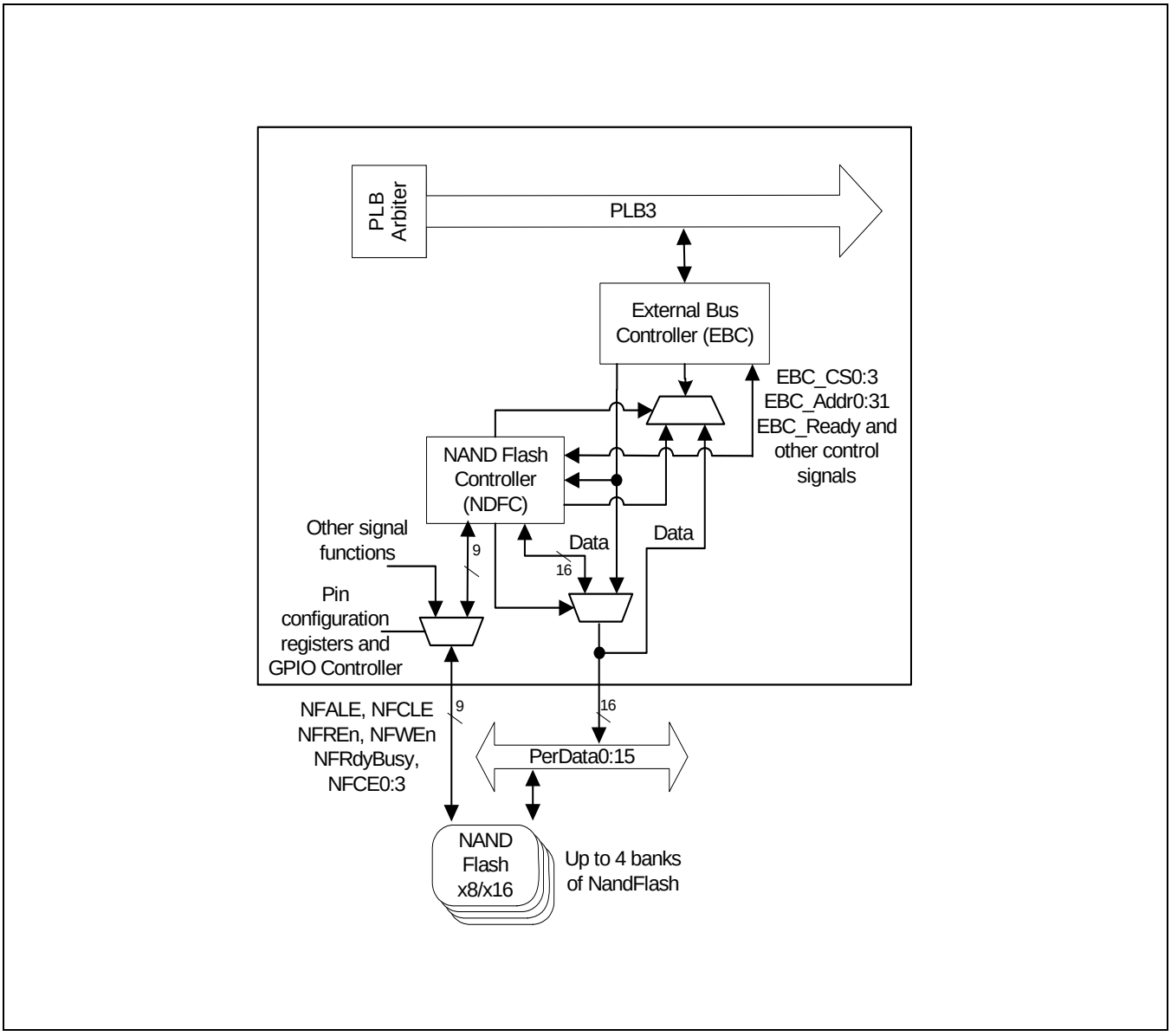
23. NAND Flash Controller

The NAND Flash controller (NDFC) enables glueless interfacing to NAND Flash.

NDFC has the following features:

- Direct interfacing to discrete NAND Flash devices (up to 4 devices)
- Device Sizes:
  - 4MB and larger supported for read/write access
- ECC generation—hamming code, single bit correction, double bit detection (SEC/DED)
- x8 wide command write, x8 wide address write, x8 and x16 bit wide data read/write
- Automatically generates RE# and WE# strobes with configurable strobe pulse width parameter
- Supports DMA in memory-to-memory copying mode to allow direct, no-processor-intervention block copy from NAND Flash out to DDR (after initial command/address setup performed)
- Interrupt on device becoming ready (after long page write or block erase operations).
- Boot-from-NAND: Execute a linear sequence of boot code out of the first 4KB of block 0.
- Operates from EBC peripheral clock.
- Frequency range for operation: Up to 83 MHz
- Class 2 power management

Figure 23-1. Typical System Application



23.1 Overview

The NDFC supports up to four NAND Flash devices. It provides direct command, address, and data access to the external device as well as a 4KB memory mapped linear access region. The 4KB linear access region enables boot code to execute up to 4KB directly from NAND flash. NDFC also provides hardware support for the Hamming code ECC algorithm that assist software in 1-bit error correction and 2-bit error detection (per 256B of data).

23.2 NDFC Signal Multiplexing

NDFC shares I/O with EBC and GPIO signals. In order to select the NDFC signals, SDR0_CUST0[MEN] must be set to 0x2. Additionally, SDR0_CUST0[NGC] must be configure to select the NDFC chip enables (NFCE0:3) as these signals are shared with the EBC (PerCS0:3). The NDFC data signals (NFData00:15) are shared dynamically with EBC (PerData00:15) depending on the access.

23.3 NDFC Interface

The NDFC interface has standard NAND flash control signals which can be directly connected to NAND flash. The NDFC data interface however uses the external EBC interface signals PerData0:15. These signals are numbered using IBM bit numbering. PerData0 is the MSb of the least significant byte. PerData8 is the msb of the most significant byte.

For 8 and 16-bit NAND flash connect PerData0:7 to NAND flash data 7:0. Commands are driven on PerData0:7 for both interface widths.

For an 8-bit NAND flash connect as follows:
PerData0 <> NAND flash data7
PerData1 <> NAND flash data6

.
.
.

PerData7 <> NAND flash data0

For a 16-bit NAND flash connect as follows:
PerData0 <> NAND flash data 7
PerData1 <> NAND flash data6

.
.
.

PerData7 <> NAND flash data0

PerData15 <> NAND flash data8
PerData14 <> NAND flash data9

.
.
.

PerData8 <> NAND flash data15

23.4 Resetting the Controller

Software should use SDR0_SRST1[NDFC] to reset the controller.

23.5 Configuring EBC to Support the NAND Flash Controller

Access to the NDFC memory mapped register is through the EBC. The base address of the NDFC registers is dependent on the memory map specified by the EBC bank registers. For each NDFC chip enable (NFCE0:3) attached to NAND flash, the corresponding EBC bank registers must be configured.

23.5.1 EBC0_BxAP

For the EBC bank or banks that are used to communicate with the NDFC, the EBC0_BxAP registers must be set to 0x018003C0.

When booting from NAND flash, the NDFC uses EBC bank 0 and therefore accesses the NDFC through the EBC using the default EBC0_B0AP configuration. After reset, EBC0_ B0AP is set to 0x7F8FFE80, which corresponds to maximum transfer timings for EBC bank 0 (TWT = 255, CSN = OEN = WBN = WBF = 3, TH=7). In addition to the wait states, ready enable is disabled, EBC0_B0AP[RE]=0. This EBC0_B0AP configuration does not provide adequate wait states or allow EBC_Ready to pace the transactions between the EBC and NDFC. To ensure the EBC does not prematurely access to the NDFC while reading the Initial Program Load, IPL, it is necessary to enable AutoRead mode at reset time by setting SDR0_SDST2[NARE]=1 (See *Figure 8.4.6* on page 202 for setting the SDR0_SDSTP:3). AutoRead Mode when enabled with EBC Configuration Complete disabled, NDFC0_CR[EBCC]=0, prevents EBC from counting wait states. In this mode the NDFC forces the EBC to supply enough wait states between accesses. (The reset default of EBC Configuration Complete is NDFC0_CR[EBCC]=0).

The Secondary Program Load (SPL) must configure the EBC0_B0AP and NDCC0_CR (set EBC0_B0AP=0x018003C0, set NDFC0_CR[EBCC]=1 and set NDFC0_CR[ARE]=0) before copying code from the NAND flash to DDR memory or on-chip memory.

23.5.2 EBC0_CFG

Set EBC0_CFG[PTD] =1. Disabled, device paced time out, forces the EBC to wait indefinitely for an internal ready signal from the NDFC.

Set EBC0_CFG[RTC] = 0b111. When RTC=0b111, Ready Timeout Counter is set to the maximum of 2048 cycles. This prevents the EBC from ending normal NAND Flash cycles prematurely.

Settings of other EBC0_CFG fields do not affect how EBC communicates with NDFC.

23.5.3 EBC0_BxCR

The NDFC registers are accessed through the EBC in the EBC memory map. The base address in the EBC0_BxCR[BAS] is the base address for the NDFC registers.

The EBC bank size can be set to the smallest size since the NDFC only requires enough memory space for the NDFC registers and a 4KB linear access region. Set EBC0_BxCR[BS]=0b000.

See *Section 23.6* for a description of the 4KB linear access region.

Set EBC0_BxCR[BU]=0b11. The NDFC registers are read/write.

Set EBC0_BxCR[BW]=0b10. The NDFC is a 32-bit EBC attached slave device.

23.5.4 Summary of EBC and NDFC Interface Configurations

1. Configure EBC0_BxCR
   - set EBC0_BxCR[BAS] to the base address desired
   - set EBC0_BxCR[BS]=0b000
   - set EBC0_BxCR[BU]=0b11
   - set EBC0_BxCR[BW]=0b10
2. Set EBC0_BxAP=0x018003C0
3. Set EBC0_CFG=0x78000000
4. Configure SDR0_CUST0

- set SDR0_CUST0[MEN]=0b10
- set SDR0_CUST0[NE]=1, to enable the NDFC controller
- Enable the NDFC chip enables, SDR0_CUST0[NCG]

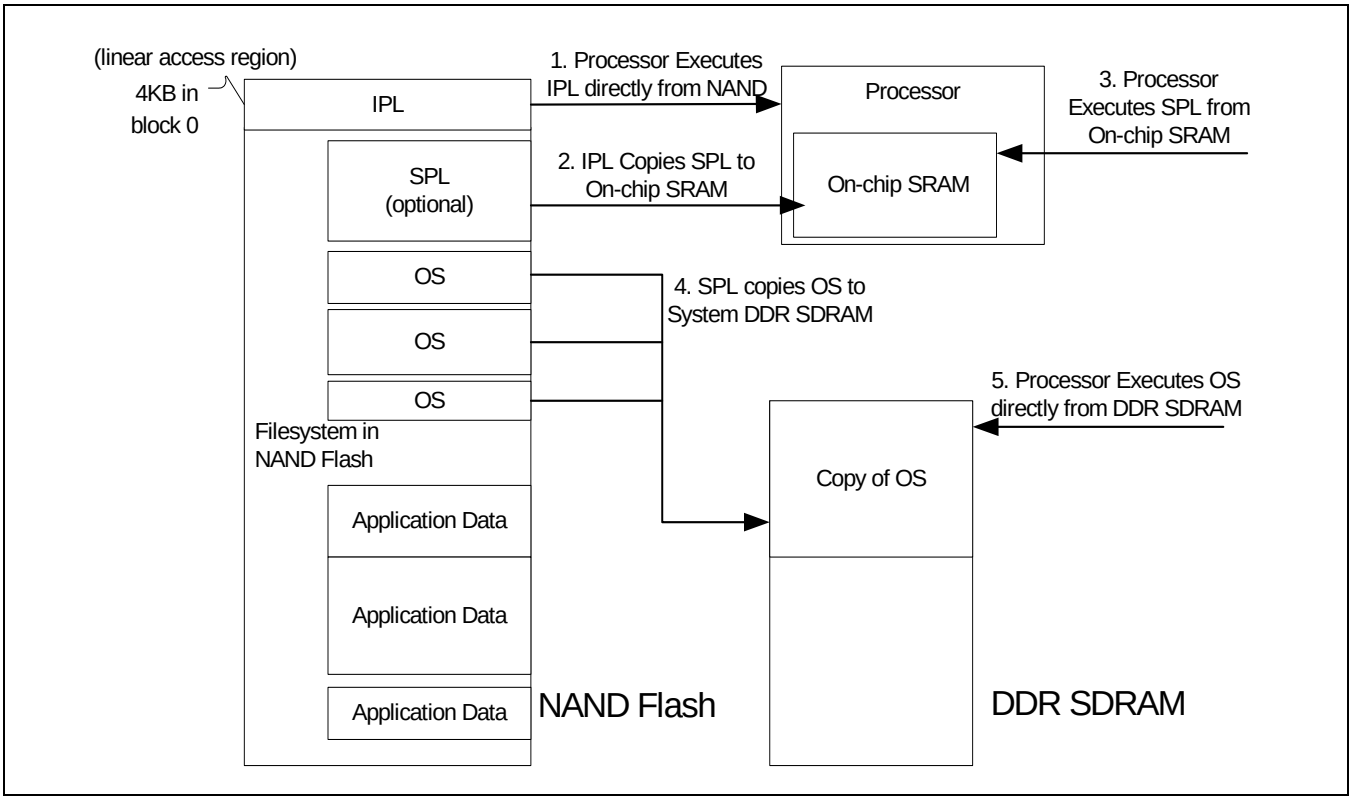
5. Configure the GPIO (See *Table 31-6* on page 803) if using $\overline{\text{NFCE1:3}}$
6. Configure NDFC0_CR and NDFC0_BxCR

23.6 Booting from NDFC

When configured for booting, the NDFC's 4KB linear access region is the target of the PowerPC boot vector access at 0xFFFFFFFC. The 4KB linear access region is mapped to the last 4KB of memory, 0xFFFE0000 - 0xFFFFFFFF, and should contain the Initial Program Loader (IPL). The linear access region is located in block 0 of the NAND flash. Typically the linear region corresponds to pages 0, 1, 2 and 3 of memory with 512B pages or pages 0 and 1 of memory with 2KB pages depending on SDR0_SDSTP2[NBP]. The first page accessed in block 0 depends on the page number pointed to by SDR0_SDSTP2[NBP].

The linear access region at boot time behaves from a software perspective as a 4KB ROM. When booting, auto-read mode is enabled, NDFC0_CR[ARE]. In this mode, the NAND controller generates the appropriate NAND flash commands and address cycles. The format of the command and address cycles used in auto-read mode is selected by NDFC0_CR[ARAC].

Figure 23-2. IPL/SPL Boot Sequence using Boot-from-NAND



Note: Block 0 must have a low error rate when booting from NAND flash. There is no way to check ECC when executing the IPL from the 4KB linear region.

In order to boot from the NAND flash, several parameters must be configured prior to booting by either Boot Option D in Table 8-3 on page 191 or Boot Option G or H. The Bootstrap Controller programs SDR0_SDSTP0:3 using the default settings of Option D or with settings read from an IIC serial ROM when using Option G or H.

Table 23-1 lists the boot configurations required for booting from NAND flash. Column 1 contains the read only SDR0_SDSTP bit fields initialized by the Bootstrap Controller. Column 2 are the configuration straps for the NDFC and EBC. The values in the SDR0_SDSTP are used to initialize the registers in column 2. Column 3 are the EBC and NDFC that are initialized by the registers in column 2.

SDR0_CUST0[MEN, NCG, NE] must be set so that the NDFC signals and the NDFC are enabled.

SDR0_CUST0[NDRSC] is a 16-bit value that specifies the maximum number of clock cycles that the NDFC waits for NFRdyBusy (1 = ready) after a device reset command, before proceeding. The value used by Boot Option D is recommended.

SDR0_CUST1[NDRDC] is a 16-bit value that specifies the maximum number of clock cycles that the NDFC waits for NFRdyBusy (1 = ready) after a page read command, before proceeding. The worst case for the page read access time in many NAND flash devices is 50μs (specified in NAND flash device data sheets as tR). The value used by Boot Option D is recommended.

Set SDR0_CP440[RL] = 0b10. RL=0b10 configures the boot source as NDFC.

Set SDR0_EBC0[RW] = 0b10. The NDFC is internally a 32 bit EBC attached slave device; therefore, RW must be set to 0b10.

Some registers in the NDFC controller itself must be configured before booting. By virtue of having selected NDFC as a boot device, the PPC440EPx/GRx automatically sets some bits in NDFC0_B0CR and NDFC0_CR. In particular, the timing values CRW, RWP, RWH, and RR are set to their maximum.

Set SDR0_SDSTP2[NBW] to the bus width of the NAND flash.

Set SDR0_SDSTP2[NBAC] to the addressing mode and page size supported by the NAND flash. This configuration is used by Auto-read mode to generate the correct address cycles when the IPL executes code from the 4KB linear region.

.

Table 23-1. Strap Register Configuration for Booting from NAND Flash

Read only Strap	Read/Write Strap	EBC and NDFC Bit Fields
SDR0_SDSTP1[RW]=0b10	SDR0_EBC0[RW]=0b10	EBC0_B0CR[BW]=0b10
SDR0_SDSTP1[RL]=0b10	SDR0_CP440[RL]=0b10	N/A
SDR0_STP2[MEN]=0b10	SDR0_CUST0[MEN]=0b10	N/A
SDR0_SDSTP2[NE]=1	SDR0_CUST0[NE]=1	NDFC0_B0CR[EN]=1
SDR0_SDSTP2[NBW]	SDR0_CUST0[NBW]	NDFC0_B0CR[SZ]
SDR0_SDSTP2[NBP]=0b000	SDR0_CUST0[NBP]=0b000	NDFC0_CR[RPG]=0b000
SDR0_SDSTP2[NBAC]	SDR0_CUST0[NBAC]	NDFC0_CR[ARAC]
SDR0_SDSTP2[NARE]=1	SDR0_CUST0[NARE]=1	NDFC0_CR[ARE]=1
SDR0_SDSTP2[NRB]=0	SDR0_CUST0[NRB]=0	NDFC0_CR[REN]=0
SDR0_SDSTP2[NDRSC]=0X8235	SDR0_CUST0[NDRSC]=0X8235	N/A
SDR0_SDSTP2[NCG]=0b1000	SDR0_CUST0[NCG]=0b1000	N/A
SDR0_SDSTP3[NDRDC]=0X0D05	SDR0_CUST1[NDRDC]=0X0D05	N/A

23.6.1 AutoRead Mode

In the AutoRead mode (NDFC0_CCR[ARE]=1), the NDFC automatically generates the page read command and page read address based on the read cycle address that is driven by the EBC. If the EBC read cycle's address is exactly +4 from the previous read cycle then the NDFC generally does not issue a new read command to the NAND Flash device, rather reads and return the next sequential piece of data. The exception to this '+4' rule is when an access crosses the NAND Flash device's data page boundary. To prevent reading data in the device's spare area and returning this as valid data, the NDFC detects when an access crosses the device's data page size boundary and also treats this access as non-sequential.

Sequential read address detection allows the NDFC to avoid unnecessary read commands, and their corresponding page read access delay (tR), for code that is fetched sequentially from memory. EBC write cycles to the NDFC do not affect the sequential read address detection, so for example, the NDFC configuration registers can be written to update the configuration of NDFC0_CR to speed up external NAND Flash access timings. Also read cycles that are not inside the linear access region do not affect the sequential read address detection, so reads of the NDFC configuration or status registers are possible as well.

Note: AutoRead Mode is only used at boot time when executing the IPL directly from NAND flash. It enables access to the 4KB linear access region.

The NDFC supports NAND Flash using AutoRead mode and Ready/Busy enabled when the device tWB is lower than

PerClk period x (NDFC0_B0CR[RWH] + 5).

In the case of boot from NAND Flash, the NDFC0_B0CR[RWH] default value is 8, which makes

tWB < (PerClk period x 13).

For PerClk = 83MHz, the device tWB must be lower than 156ns

For PerClk = 66MHz, the device tWB must be lower than 195ns

After boot, for better performance, the NDFC0_B0CR[RWH] value can be adjusted to be the first value immediately higher than tWB.

In auto-read mode, the number of command and address cycles is dependent on the NDFC0_CR[ARAC] configuration. *Table 23-2* contains the command and address sequence for each auto-read access cycle configuration. The commands and address driven on the NDFC data interface are described in Tables *23-3* and *23-4*. *Table 23-5* indicates how the row and column address cycles relate to the linear access region and the internal EBC address bus.

Table 23-2. Address/Commands Sequence Auto-Read Mode

SDR0_SDSTP[NBAC] SDR0_CUST0[NBAC] NDFC0_CR[ARAC]	Page Size	# CMD Cycles	# Address cycles	Order of Cycles
00	512B	1	3 cycles	<CMD0>,<column0>,<row0>,<row1>
01	512B	1	4 cycles	<CMD0>,<column0>,<row0>,<row1>,<row2>
10	2KB	2	4 cycles	<CMD0>,<column0>,<column1>,<row0>,<row1>,<CMD1>
11	2KB	2	5 cycles	<CMD0>,<column0>,<column1>,<row0>,<row1>,<row2>,<CMD1>

Table 23-3. Page Read Commands for an Auto-Read Cycle

SDR0_SDSTP[NBAC] SDR0_CUST0[NBAC] NDFC0_CR[ARAC]	Device Width	Page Size	<CMD0>	<CMD1>	Description
00, 01	8-bit	512B	0b0000000N	n/a	The N in 0b0000000N <CMD0> is driven by EBC_Addr23. To select either the 1st or 2nd 256-Bytes in a 512- Byte page size device, the distinction between a READ0 and a READ1 command is made with internal EBC_Addr23.
	16-bit	512B	0x00	n/a	A 16-bit wide device with a 512-Byte data page size corresponds to 256 halfword-sized columns, which can be all be reached with 1 column access cycle, and thus the READ1 command is unneces-sary
10, 11	8/16-bit	2KB	0x00	0x30	2kB data page size devices are assumed to require a 2nd read command input after the address cycles to start the page read access in the device.

Table 23-4. Page Address for an Auto-Read Cycle

Cycle	Device Width	Page Size	<Address>	Description
<column0>	8-bit	512B/2KB	PerAddr24:31	LS byte address. Address driven internally to NAND flash controller on EBC address bits EBC_Addr24:31.
	16-bit	512B/2KB	PerAddr23:30	LB halfword address. Address driven internally to NAND flash controller on EBC address bits EBC_Addr23:31.
<column1>	8-bit	2KB	0b00000NNN	MS byte address for a 2KB page size device. NNN is driven internally to NAND flash controller by EBC address bits EBC_Addr21:23.
	16-bit	2KB	0b000000NN	MS byte address for a 2KB (1K-halfwords) page size device. NN is driven internally to NAND flash controller by EBC address bits EBC_Addr21:22.
<row0>	8/16-bit	512B	0b0PPPPNNN	select one of eight 512B pages in block m. PPPP driven with the page number specified by the bootstrap SDR0_SDSTP2[NBP]. NNN is driven internally to NAND flash controller by EBC address bits EBC_Addr20:22.
		2KB	0b000PPPPN	select one of two 2KB pages in block m. PPPP driven with the page number specified by the bootstrap SDR0_SDSTP2[NBP]. N is driven internally to NAND flash controller by EBC address bits EBC_Addr20.
<row1>	8/16-bit	512B/2KB	0b00000000	
<row2>	8/16-bit	512B/2KB	0b00000000	

Table 23-5. Address Driven with Auto-Read Mode Enabled

Internal EBC address EBC_Addr0:31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NDFC Linear Access Region																				1	Linear Access Region											
NAND Device Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 23-5. Address Driven with Auto-Read Mode Enabled

512B page 8-bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PPPP	Row0		a*	Column0
512B page 16-bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PPPP	Row0			Column0 -
2KB page 8-bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PPPP	R0	Col1		Column0
2KB page 16-bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		PPPP	R0	Col1		Column0 -

PPPP driven with the page number specified by the bootstrap SDR0_SDSTP2[NBP].
a* is driven by EBC_Addr23. To select either the 1st or 2nd 256-Bytes in a 512- Byte page size device, the distinction between a READ0 and a READ1 command is made with internal EBC_Addr23.

23.6.2 Accessing NAND Flash

Access to NAND flash is through the NDFC0_CMD, NDFC0_ADDR and NDFC0_DATA registers. Writing NDFC0_CMD and NDFC0_ADDR registers generates command and address cycles and reading or writing the NDFC0_DATA register generates command and addres cycles and reading or writing to the NDFC0_DATA register generates data access on the NDFC interface. The command written on the NDFC interface is the same as the value written to the NDFC0_CMD register. Refer to the NAND Flash data sheet for command and address cycles required for a particular NAND flash.

23.6.2.1 Reading NAND Flash

The following example is a NAND flash read and demonstrates four address cycles:

```
    outbyte(NDFC0_CMD, command);
    outbyte(NDFC0_CMD, command);    //read command
    outbyte(NDFC0_ADDR, first_addr);
    outbyte(NDFC0_ADDR, second_addr);
    outbyte(NDFC0_ADDR, third_addr);
    outbyte(NDFC0_ADDR, fourth_addr);

//  Wait for address cycles to complete

    for (i=0; i < size; i++) // Size is the number of bytes in the page
    {
         destinationAddress[addr]=(inbyte(NDFC0_DATA));
         addr++;
     }
```

23.6.2.2 Writing NAND Flash

The following example is a NAND flash write and demonstrates four address cycles:

```
    outbyte(NDFC0_CMD, command);
    outbyte(NDFC0_CMD, command);    //write command
    outbyte(NDFC0_ADDR, first_addr);
    outbyte(NDFC0_ADDR, second_addr);
    outbyte(NDFC0_ADDR, third_addr);
    outbyte(NDFC0_ADDR, fourth_addr);

//  Wait for address cycles to complete
```

```
for (i=0; i < size; i++) // Size is the number of bytes in the page
{
   outbyte(NDFC0_DATA , sourceAddress[i];
}
```

23.7 ECC

The NDFC implements a Hamming Code Error Correction Code (ECC) that is compatible with SmartMedia format. This ECC is based on parity calculations of the read (incoming) or write (outgoing) data and consists of 3 Bytes (22-bits) per 256 Byte (2048-bits) data segment and provides single-bit error correction, double-bit error detection (SEC/DED) on the 256 Byte block. The 22-bit ECC code is composed of 16-bit line parity and 6 bit column parity.

- Hamming ECC Algorithm is directly compatible with SmartMedia specification.
- ECC accumulated during page write and is available to software for programming after either 256 or 512 Bytes of page data write cycles are completed.
- ECC calculated during page read on incoming data and is available for comparison with ECC stored in device.
- Eight ECC registers accumulate calculated ECC data for each 256 Byte block in an entire 2KB page-sized device.
- ECC hardware assist is mainly useful when reading entire pages.
- Software is responsible for making the stored versus calculated ECC comparison and performing any required bit error correction to the data block.
- The hardware ECC may be supplemented with stronger software ECC algorithm to improve coverage for Multi-Level-Cell (MLC) NAND Flash devices, where the failure of a single transistor can cause the loss of 2-bits of data.

For the SmartMedia format, the ECC code stored in the external NAND Flash device is always stored in the format shown in *Table 23-6*.

Table 23-6. ECC Code Assignment Table

I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
P64	P64'	**P32**	P32'	**P16**	P16'	**P8**	P8'
P1024	P1024'	**P512**	P512'	**P256**	P256'	**P128**	P128'
P4	P4'	**P2**	P2'	**P1**	P1'	1	1
P8 to P1024 = Line/Row Parity P1 to P4 = Column Parity							

The NDFC algorithm for generating the 22-bit ECC is compatible with the requirements for SmartMedia devices. This algorithm handles the correction of single bit errors (correctable) and the detection of double bit errors (uncorrectable).

Note that if there are 3 or more bit errors in a page of read data, then the Hamming code may improperly determine that there is a single, correctable bit error and thus indicate that the software does a bit correction. This results in 'mis-correction' of already bad data. If the application uses a NAND Flash device where more than 2 bit errors per 256-Bytes of page data is a possibility, then the software for that application should implement a stronger ECC algorithm, similar to Reed/Solomon code.

ECC is accumulated for every data byte that is either read or written via the NDFC0_DATA register or via the linear access region. After 256-Bytes of data have been read or written, the calculated ECC result is store in one of the NDFC0_ECCn registers (NDFC0_ECC0-NDFC0_ECC7) and the next NDFC0_ECCn register is selected. Then subsequent data reads/writes start accumulating a new ECC value and after 256 Bytes more data, that next result is stored and the next NDFC0_ECCn register is again selected. This process repeats until a calculated ECC result is stored into NDFC0_ECC7, which indicates that 2KB of data has been written (without any ECC reset occurring). After NDFC0_ECC7 is update, the ECC calculation is halted for subsequent reads/writes, and NDFC0_ECC0-7 registers are not updated again until an 16-bit wide NAND Flash device (NDFC0_CR[SZ]=1) causes two bytes of 'accumulation' for the ECC calculations.

To check the ECC on a block of read data is a simple matter of performing a bitwise 'XOR' function between the calculate ECC (read from NDFC0_ECC0-NDFC0_ECC7) and the store ECC (read from spare/extended data area of external NAND Flash device). The results of this XOR gives the ECC syndrome.

Table 23-7. Hamming ECC Syndromes

XOR Result	Meaning
22-bits all 0	No Error
Exactly 11 bits are 1	Correctable Error
1 bit is 1	ECC Code Error
Anything Else	Uncorrectable Error

No Error

When the XOR result value has all 22-bits with the value '0', this indicates that the stored ECC which was read from the external NAND Flash device and the calculated ECC read from the NFECC0-NFECC7 register are identical and there is not error in the page data and no error in the stored ECC data.

Correctable Errors

For correctable errors, the XOR result value has the property where all parity bit pairs (e.g. P8 and P8) have opposite values (1 & 0 or 0 & 1), with exactly 11-bits of the 22-bit code set to 1. By extracting the XOR result's parity bits (P1024, P512, P256, P128, P64, P32, P16, P8, P4, P2, P1), the exact bit position of the single bit error can be determined.

(P1024, P512, P256, P128, P64, P32, P16, P8)=Offset of byte containing single bit error (P4, P2, P1)=Bit number of single bit error.

> Example: If XOR Results is (**P1024**, P1024, **P512**, P512, ..., **P2**, P2, **P1**, P1)=22'b**0**1**1**0**0**1**0**1**1**0**1**0**1**0**0**1**0**1**1**0**1**0**0**1, this result has the correct properties to qualify as a single, correctable bit error. Extracting the parity bits (**P1024**, **P512**, ...**P16, P8**) (**P4, P2, P1**)=(01001100) (110). This result means that at byte offset 76 (4Ch), bit 6 should be flipped to correct the single bit error.

ECC Code Errors

A single bit set to '1' in the XOR result value indicates that the single bit error occurred in the stored ECC code and not in the page of read data.

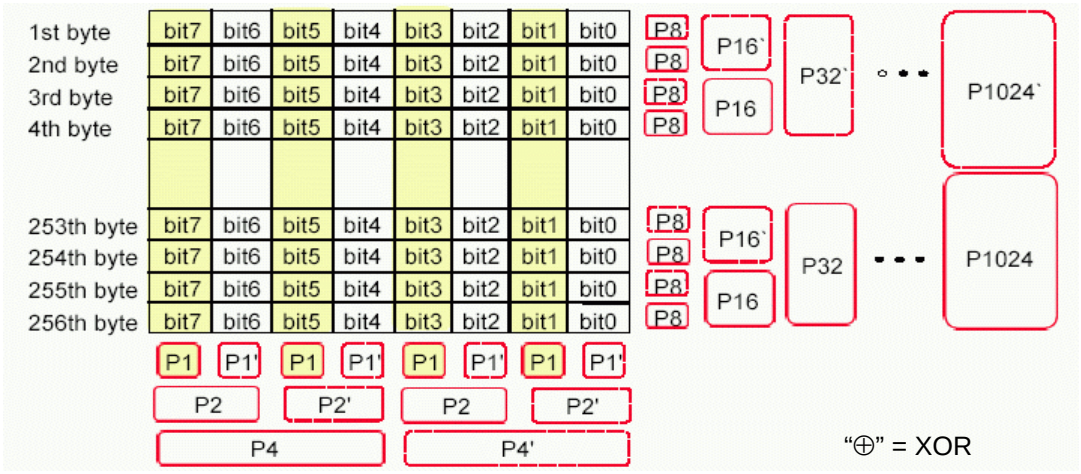
Uncorrectable Errors

If the XOR result value is non-zero and does not meet the criteria for a correctable error or ECC code error, then the error should be considered uncorrectable as two or more bits of the page data were corrupted and the Hamming ECC algorithm cannot determine which bytes/bits are faulty. In this case, the entire page of read data should be discarded.

23.7.1 Column Parity

The NDFC ECC generator calculates 6-bits of Column parity. These parity bits are called CP00 - CP05 and correspond to (P1, P1', P2, P2', P4, P4').

Figure 23-3. Column Parity Generation



P1 = bit7 ⊕ bit5 ⊕ bit3 ⊕ bit1 ⊕ P1
P2 = bit7 ⊕ bit6 ⊕ bit3 ⊕ bit2 ⊕ P2
P4 = bit7 ⊕ bit6 ⊕ bit5 ⊕ bit4 ⊕ P4

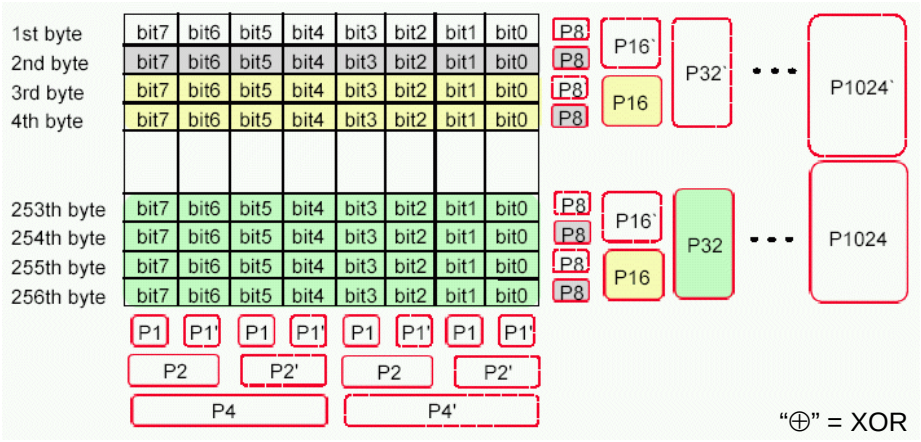
P1' = bit6 ⊕ bit4 ⊕ bit2 ⊕ bit0 ⊕ P1'
P2' = bit5 ⊕ bit4 ⊕ bit1 ⊕ bit0 ⊕ P2'
P4' = bit3 ⊕ bit2 ⊕ bit1 ⊕ bit0 ⊕ P4'

Column parity is accumulated for each byte of data that is read or written to the external NAND Flash device when ECC is enabled.

23.7.2 Line/Row Parity

The NDFC ECC generator calculates 16-bits of Line/Row parity. These parity bits are called LP00 - LP15 and correspond to (P8, P8', P16, P16', P32, P32', ... P1024, P1024').

Figure 23-4. Line/Row Parity Generation



P8 = (even byte) = bit7 ⊕ bit6 ⊕ bit5 ⊕ bit4 ⊕ bit3 ⊕ bit2 ⊕ bit1 ⊕ bit0 ⊕ P8
P8' = (odd byte) = bit7 ⊕ bit6 ⊕ bit5 ⊕ bit4 ⊕ bit3 ⊕ bit2 ⊕ bit1 ⊕ bit0 ⊕ P8'
etc...

23.7.3 How to generate line parity (LP00-LP15)

The line parity, which is recalculated for each data byte, depends on the data bytes which have already been input. If the total sum of the bits of 1 byte of data is 0, the line parity does not change when it is recalculated.

The sum of the bits in 1 byte of data is: (⊕ = XOR function)

D_{all} = D0 ⊕ D1 ⊕ D2 ⊕ D3 ⊕ D4 ⊕ D5 ⊕ D6 ⊕ D7

Sixteen line parity bits need to be computed (LP0-LP15) from 256 bytes of data. This is done by initializing all line parity bits to zero, reading in each byte, computing the byte sum bit (D_{all}), and adding D_{all} to the line parity bits when they apply. This is done by using an 8 bit counter and using the appropriate bit as a type of mask. The counter is incremented by 1 for each new byte of data.

LP00=LP00 ⊕ (¬Counter_bit0) ∧ (D_{all})	LP01=LP01 ⊕ (Counter_bit0) ∧ (D_{all})
LP02=LP02 ⊕ (¬Counter_bit1) ∧ (D_{all})	LP03=LP03 ⊕ (Counter_bit1) ∧ (D_{all})
LP04=LP04 ⊕ (¬Counter_bit2) ∧ (D_{all})	LP05=LP05 ⊕ (Counter_bit2) ∧ (D_{all})
LP06=LP06 ⊕ (¬Counter_bit3) ∧ (D_{all})	LP07=LP07 ⊕ (Counter_bit3) ∧ (D_{all})
LP08=LP08 ⊕ (¬Counter_bit4) ∧ (D_{all})	LP09=LP09 ⊕ (Counter_bit4) ∧ (D_{all})
LP10=LP10 ⊕ (¬Counter_bit5) ∧ (D_{all})	LP11=LP11 ⊕ (Counter_bit5) ∧ (D_{all})
LP12=LP12 ⊕ (¬Counter_bit6) ∧ (D_{all})	LP13=LP13 ⊕ (Counter_bit6) ∧ (D_{all})
LP14=LP14 ⊕ (¬Counter_bit7) ∧ (D_{all})	LP15=LP15 ⊕ (Counter_bit7) ∧ (D_{all})

⊕ = XOR
¬ = NOT
∧ = AND

For example, LP00 should be calculated from all even data bytes and LP01 should be calculated from all odd data bytes. Bit 0 of the counter is ANDed with D_{all} to enable or disable whether D_{all} contributes to the line parity summation.

Thus, according to the line parity generation formulas as:

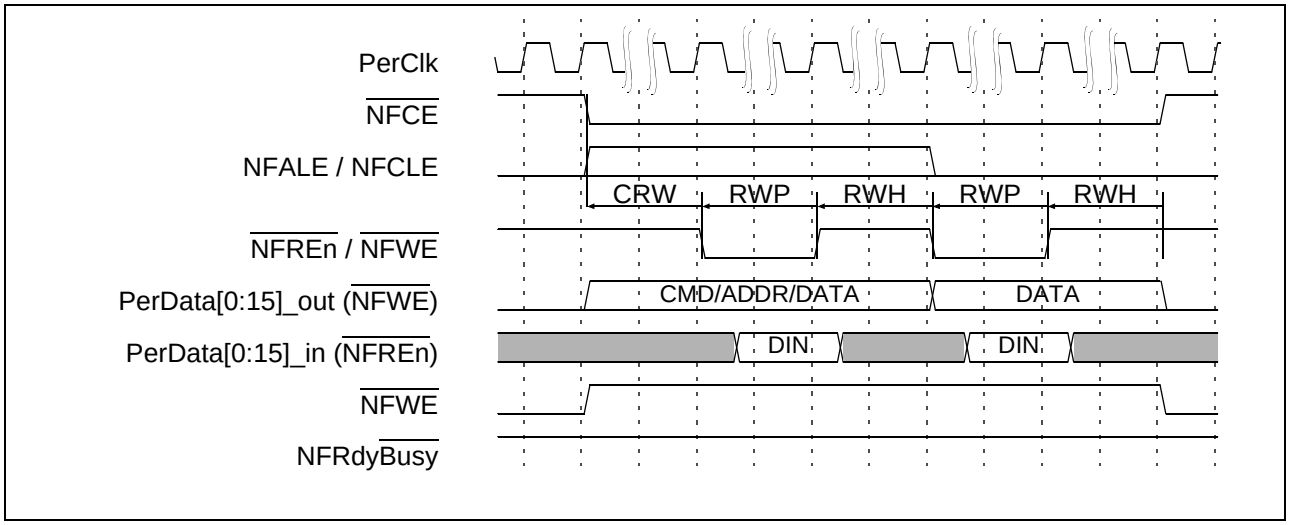
for LP00: 1st,3rd,5th,7th ... data bytes used (counter = 0, 2, 4, 6...)
for LP01: 2nd,4th,6th,8th ... data bytes used (counter = 1, 3, 5, 7...)
for LP02: 1st,2nd,5th,6th ... data bytes used (counter = 0, 1, 4, 5 ...)
for LP03: 3rd,4th,7th,8th ... data bytes used (counter = 2, 3, 6, 7...)
etc.

23.8 External Device Cycles

The bus timing on the NAND flash interface is programmable on a per chip enable basis using the timing parameters in the NFxCR registers. Figures 23-5 through 23-11, show how the timing parameters CRW, RWP, RWH affect timing of various NAND flash accesses.

- The NFALE and NFCLE outputs are active for CRW+RWP+RWH cycles as shown in Figure 23-5.
- When both NFALE and NFCLE are low, DATA cycle can continue for additional (RWP+RWH) cycles until the transfer is satisfied. For example, a 32-bit word transfer from the EBC results in (CRW)+4*(RWP+RWH) cycles on the external bus. $\overline{\text{NFREn}}$ or $\overline{\text{NFWE}}$ are active only during each of the RWP cycle times.

Figure 23-5. Basic External Device Timing Controls



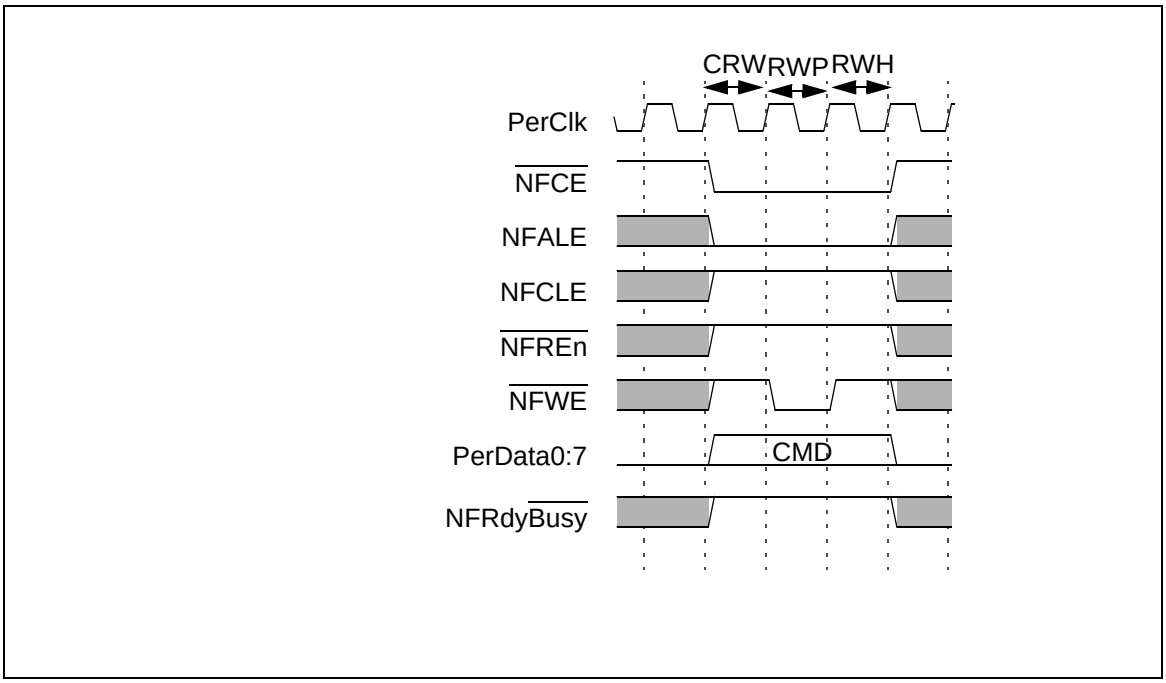
NFxCR[CRW] 0 - 7 Cycles clocks between $\overline{\text{NFCE}}$/NFALE/NFCLE asserted and $\overline{\text{NFREn}}$ / $\overline{\text{NFWE}}$ asserted. When CRW=0, $\overline{\text{NFREn}}$ / $\overline{\text{NFWE}}$ are asserted at the same time as $\overline{\text{NFCE}}$/NFALE/NFCLE.

NFxCR[RWP] 1 - 8 Cycles Clocks that $\overline{\text{NFREn}}$ / $\overline{\text{NFWE}}$ stay active (Active Pulse width).

NFxCR[RWH] 1 - 8 Cycles Clocks that $\overline{\text{NFREn}}$ / $\overline{\text{NFWE}}$ stay inactive (Data Hold time).

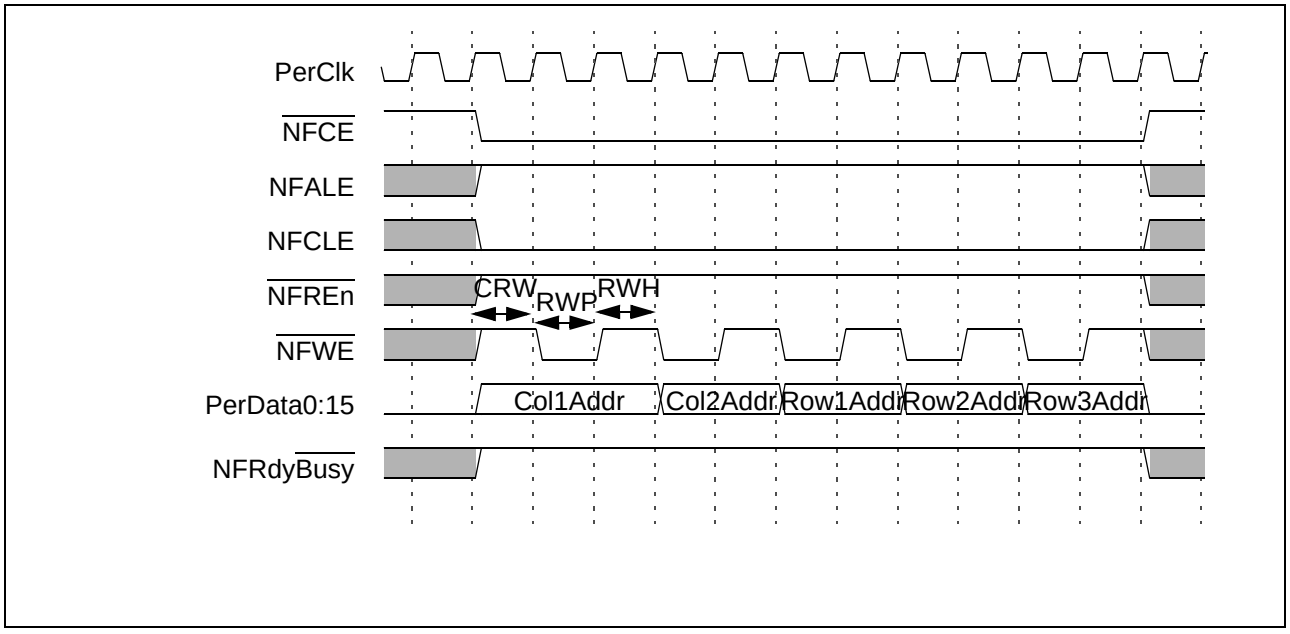
23.8.1 Command Cycle

Figure 23-6. Command write cycle



23.8.2 Address Write Cycle

Figure 23-7. Address Cycle



23.8.3 Basic Nand Flash Device Read Cycle

Figure 23-8. Single (raw) Read Cycle

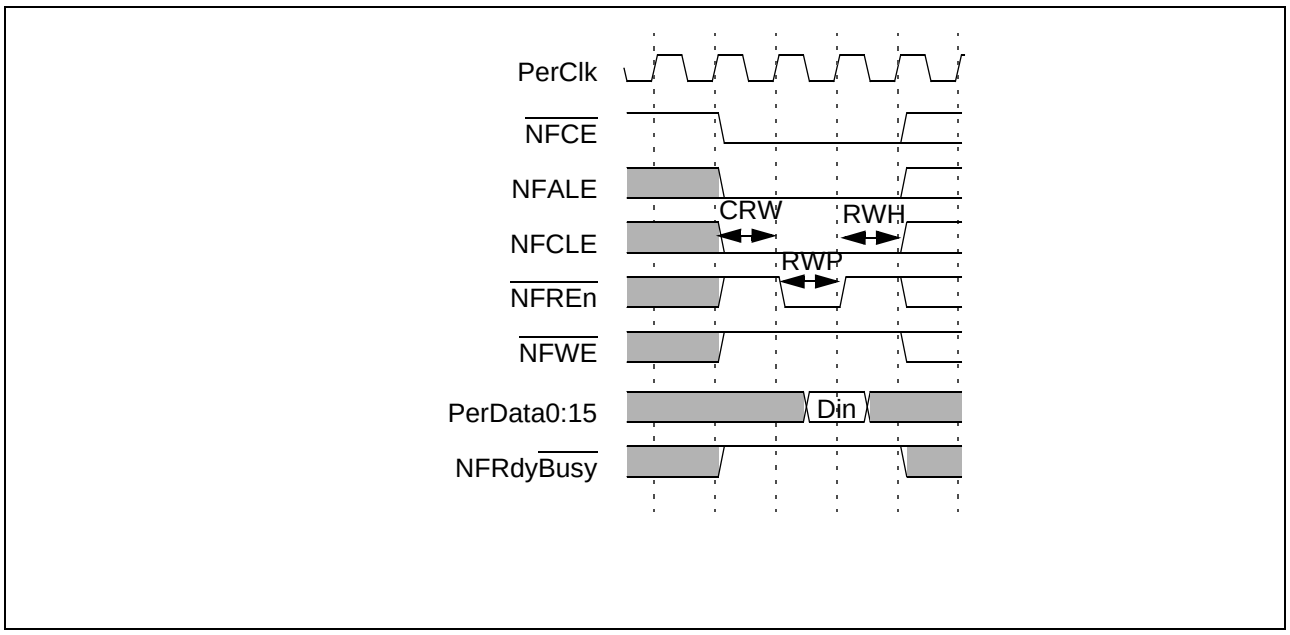
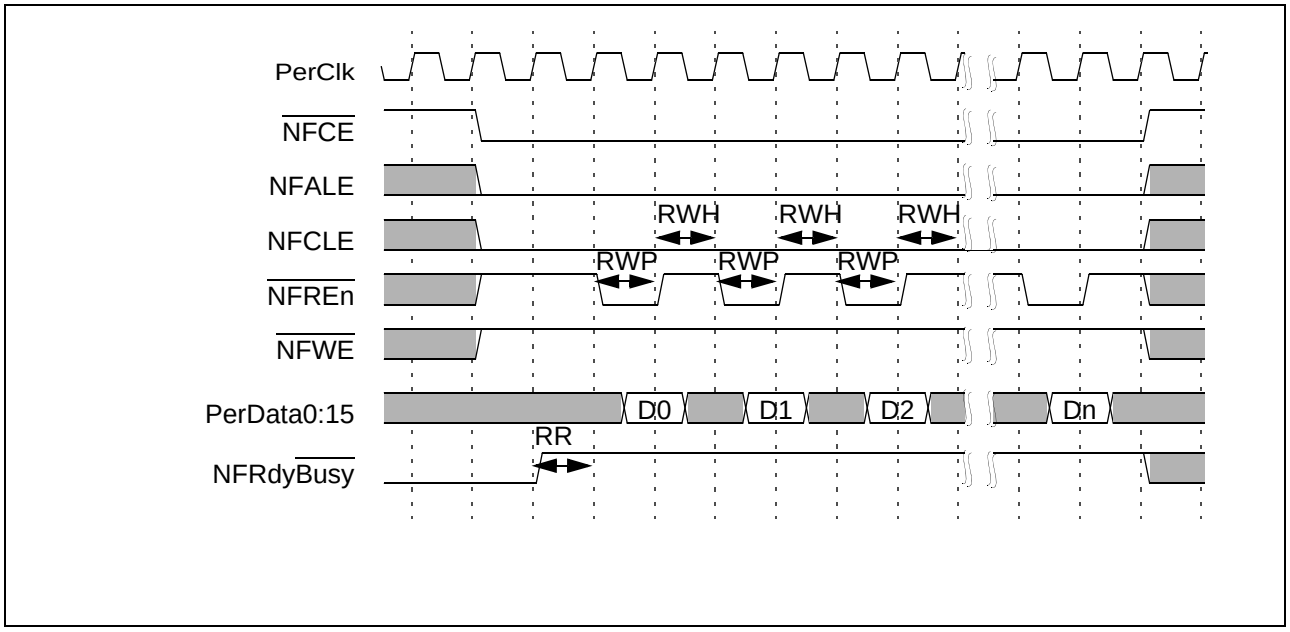


Figure 23-9. Burst (Page) Read Cycle



23.8.4 Basic Nand Flash Device Write Cycle

Figure 23-10. Single (raw) Write Cycle

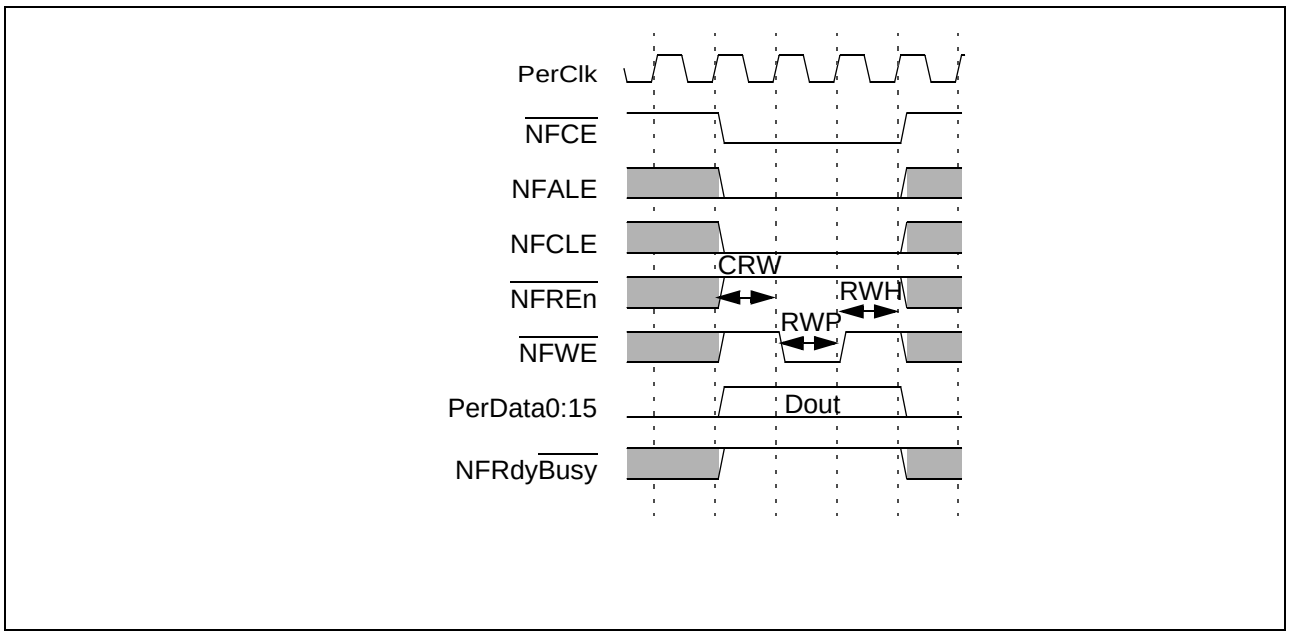
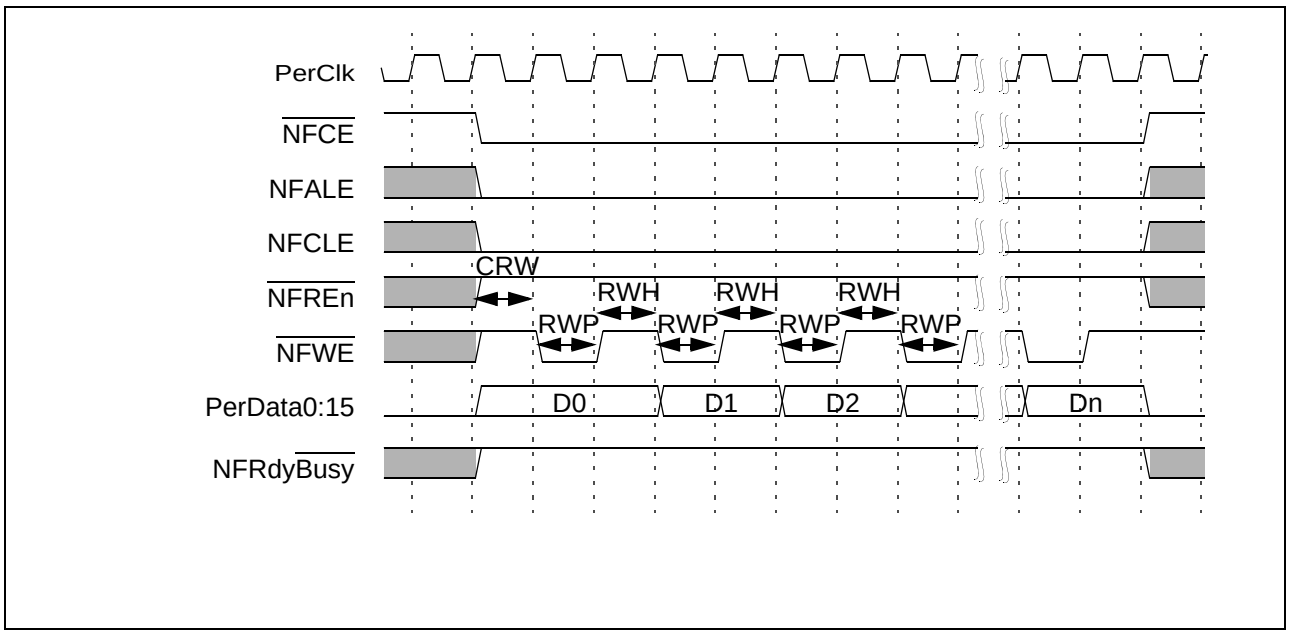


Figure 23-11. Burst (Page) Write Cycle



23.9 NDFC Registers

The NAND Flash controller provides one dedicated configuration register for each NAND Flash device chip selected (NFCE0:3). There are also 15 other common registers that are shared across all devices and are used to configure or monitor status of the common aspects of the NDFC.

23.9.1 Memory Map

This section lists all of the NDFC registers and describes how they are accessed.

The NDFC is relocatable within the EBC address space. See *External Bus Controller* on page 555 for details on how to program the base address and region size and assign the NDFC to one of the EBC banks. Addresses in *Table 23-8* are offsets to the EBC base address, EBC0_BnCR[BAS]. For example EBC0_B1CR[BAS]=0x900, software accesses the NDFC0_ADDR register at PLB3 address 0x90000004. On PLB4, the 36 bit address for NDFC0_ADDR is 0x190000004.

Table 23-8. NAND Flash Controller Memory Map

Mnemonic	Register	Address Offset	Access	Page
NDFC0_CMD	NDFC Command Register	0x0000	R/W	601
NDFC0_ADDR	NDFC Address Register	0x0004	R/W	601
NDFC0_DATA	NDFC Data Register	0x0008	R/W	601
NDFC0_ECC0	NDFC ECC Register 0	0x0010	R	603
NDFC0_ECC1	NDFC ECC Register 1	0x0014	R	603
NDFC0_ECC2	NDFC ECC Register 2	0x0018	R	603
NDFC0_ECC3	NDFC ECC Register 3	0x001C	R	603
NDFC0_ECC4	NDFC ECC Register 4	0x0020	R	603
NDFC0_ECC5	NDFC ECC Register 5	0x0024	R	603
NDFC0_ECC6	NDFC ECC Register 6	0x0028	R	603
NDFC0_ECC7	NDFC ECC Register 7	0x002C	R	603
NDFC0_B0CR	NDFC Bank Configuration Register 0	0x0030	R/W	603
NDFC0_B1CR	NDFC Bank Configuration Register 1	0x0034	R/W	603
NDFC0_B2CR	NDFC Bank Configuration Register 2	0x0038	R/W	603
NDFC0_B3CR	NDFC Bank Configuration Register 3	0x003C	R/W	603
NDFC0_CR	NDFC Configuration Register	0x0040	R/W	604
NDFC0_SR	NDFC Status Register	0x0044	R	606
NDFC0_HWCTL	NDFC Direct Hardware Control Register	0x0048	R/W	607
NDFC0_REVID	NDFC Revision ID Register	0x0050	R	608
na	Linear Data Access Region	0x1000 - 0x1FFF	R/W	na

23.9.2 NDFC Address Register (NDFC0_ADDR)

The NAND Flash Address register (NDFC0_ADDR) is an external hardware access register. When a write is made to this register, the NFALE signal is asserted and a write cycle is performed to the external device.

External device cycles are only performed when the currently selected bank, as programmed in the NDFC0_CR[BS] field, is enabled in its bank configuration register, NDFC0_CR[EN]. When the currently selected bank is not enabled, writes and reads to this register simply store and return stored data, respectively.

This register can be written with a transfer size of byte, half word or word.

Reads of this register only return the data stored in the NDFC0_ADDR register. No external device cycle is performed for a read of this register. Unused bits return a value of '0'

Figure 23-12. NAND Flash Address Register (NDFC0_ADDR)

0:7	ADDR	Nand Flash Command	
8:31		Reserved	

23.9.3 NDFC Command Register (NDFC0_CMD)

The NAND Flash Command register (NDFC0_CMD) is an external hardware access register. When a write is made to this register, the NFCLE signal is asserted and a write cycle is performed to the external device.

External device cycles are only performed when the currently selected bank (as programmed in the NDFC0_CR[BS] field) is enabled in its bank configuration register, NDFC0_BnCR[EN]. When the currently selected bank is not enabled, writes and reads to this register simply store and return stored data, respectively.

This register can be written with a transfer size of byte, half word or word.

Reads of this register only return the data stored in the NDFC0_CMD register. No external device cycle is performed for a read of this register. Unused bits return a value of 0.

Figure 23-13. NAND Flash Command Register (NDFC0_CMD)

0:7	CMD	Nand Flash Command	
8:31		Reserved	

23.9.4 NDFC Data Register (NDFC0_DATA)

The NAND Flash Data register (NDFC0_DATA) is an external hardware access register. When a write is made to this register, one or more write cycles are performed to the external device. When a read is made to this register, one or more read cycles are performed to the external device.

External device cycles are only performed when the currently selected bank, as programmed in the NDFC0_CR[BS] field, is enabled in its bank configuration register, NDFC0_CR[EN]. When the currently selected bank is not enabled, writes and reads to this register simply store and return stored data, respectively.

This register can be written with a transfer size of byte, half word or word.

Note that a byte size read transfer to a 16-bit NAND Flash device (NDFC0_CR[SZ]=1) results in the loss of the upper half of the data returned from the device. This condition is detected and causes the NDFC0_SR[RDL] bit to be set.

The total number of external accesses performed for each transfer size is dependent on the selected bank's device width.

Figure 23-14. NAND Flash Data Register (NDFC0_DATA)

0:31	DATA	Nand Flash Data	

Table 23-9 and *Table 23-10* describe the data input and out ordering, respectively.

Table 23-9. Data Input Ordering to Read Data on EBC

Device Size	Read size	External Cycle #	PerData0:7	PerData8:15	Internal EBCdata bus (0:31)[a]
16	Word (4 bytes)	1 2	<aa> <cc>	<bb> <dd>	<**aabbccdd**>
	Half Word (2 bytes)	1	<aa>	<bb>	<**aabb**aabb>
	Bytes (1)[b]	1	<aa>	<bb>	<**aa**aaaaaa>
8	Word (4 bytes)	1 2 3 4	<aa> <bb> <cc> <dd>	<XX> <XX> <XX> <XX>	<**aabbccdd**>
	Half Word (2 bytes)	1 2	<aa> <bb>	<XX> <XX>	<**aabb**aabb>
	Bytes (1)	1	<aa>	<XX>	<**aa**aaaaaa>

a. Reads of less than a word returns data mirrored onto all possible byte lanes.
b. A byte size read from a 16-bit device results in loss of data returned from the device. This sets the NDFC0_STAT[RDL] flag.

Table 23-10. Write Data to Output Ordering

Device Size	Internal EBCdata bus(0:31)	Write size	External Cycle #	PerData0:7	PerData8:15
16	<aabbccdd>	Word (4 bytes)	1 2	<aa> <cc>	<bb> <dd>
	<aabb>	Half Word (2 bytes)	1	<aa>	<bb>
	<aa>	Bytes (1)	1	<aa>	<XX>
8	<aabbccdd>	Word (4 bytes)	1 2 3 4	<aa> <bb> <cc> <dd>	<ZZ> <ZZ> <ZZ> <ZZ>
	<aabb>	Half Word (2 bytes)	1 2	<aa> <bb>	<ZZ> <ZZ>
	<aa>	Bytes (1)	1	<aa>	<ZZ>

23.9.5 NAND Flash ECC Calculation Registers (NDFC0_ECC0:NDFC0_ECC7)

The NAND Flash ECC Calculation Registers (NDFC0_ECC0:NDFC0_ECC7) are for reading the hardware generated ECC values for each 256-Bytes of page read or page write data.

These registers cannot be written. They can be cleared using the NDFC0_CR[ECR] (ECC Reset) bit. Reads of the NDFC0_ECC0:7 registers only return the ECC data stored in the register and do not affect the calculation or delete the ECC data.

Figure 23-15. NAND Flash ECC Calculation Register (NDFC0_ECC0:NDFC0_ECC7)

0:7	BC	Byte Counter	
8:15	LP0	Line/Row Byte 0 ECC Calculated Result 8 - P64 9 - P64' 10 - P32 11 - P32' 12 - P16 13 - P16' 14 - P8 15 - P8 '	
16:23	LP1	Line/Row Byte 1 ECC Calculated Result 16 - P1024 17 - P1024' 18 - P512 19 - P512' 20 - P256 21 - P256' 22 - P128 23 - P128'	
24;31	CP	Column (bit offset) Calculated Result 24 - P4 25 - P4' 26 - P2 27 - P2' 28 - P1 29 - P1' 30 - 1 31 - 1	

23.9.6 NDFC Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR)

The NAND Flash Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR) are for enabling and setting access parameters for each of the external banks of NAND Flash.

This register can be written with a transfer size of byte, half word or word.

Figure 23-16. NAND Flash Bank Configuration Registers (NDFC0_B0CR:NDFC0_B3CR)			
0	EN	Bank Enable 0 Bank access disabled 1 Bank access enabled	When the NDFC0_ADDR, NDFC0_CMD, NDFC0_DATA, or Linear Data Access Region are accessed and the currently selected bank is disabled, the NDFC will not perform an external bus cycle, but rather just store the data into the appropriate register. NDFC0_CR reset default comes from SDR0_SDST2[NE].
1	CED	CE# style 0 CE# will remain asserted/active until ECC Reset or switching bank selects via NDCF0_CR 1 CE# is 'Don't Care' style and will deassert when external access cycle is completed	When cleared, the selected bank's CE# signal is asserted when the first access is made to NDFC0_CMD, NDFC0_ADDR, or NDFC0_DATA registers or the Linear Data regions and will remain asserted until a CE# reset is performed (NDFC0_CR[CER]) or the active bank is switched (NDFC0_CR[BS]). NDFC0_CR reset default is always 0.
2:3		Reserved	
4	SZ	Bank Size 0 External Device is 8-bit, attached to PerData[00:07] 1 External Device is 16-bit, attached to PerData[00:15]	If SDR0_SDST2[NE]=1, NDFC0_CR reset default comes from SDR0_SDSTP2[NBW].
5:16		Reserved	
17:19	CRW	CE# low to RE#/WE# starting (0-7)	Setting is based on PerClk cycles If SDR0_SDST2[NE]=1, NDFC0_CR reset default=0b111.
20		Reserved	
21:23	RWP	RE#/WE# low pulse time (1-8)	Setting is based on PerClk cycles If SDR0_SDST2[NE]=1, NDFC0_CR reset default=0b111.
24		Reserved	
25:27	RWH	RE#/WE# high pulse time (1-8)	Setting is based on PerClk cycles If SDR0_SDST2[NE]=1, NDFC0_CR reset default=0b111.
28		Reserved	
29:31	RR	RDY_BSY# high to RE# starting (1-8)	Setting is based on PerClk cycles If SDR0_SDST2[NE]=1, NDFC0_CR reset default=0b111. Note: This value only has an effect for read accesses in the linear region and only when NDFC0_CR[REN] is set.

23.9.7 NDFC Configuration Register (NDFC0_CR)

The NAND Flash Configuration Register (NDFC0_CR) is for enabling and setting access parameters for the entire NDFC.

This register can be written with a transfer size of byte, half word or word.

Figure 23-17. NAND Flash Configuration Register (NDFC0_CR)			
0	CER	CE# Reset 0 no action 1 Reset any active Chip Enable output	Setting this bit to 1 deasserts an active CE# output. This is only necessary when a bank is configured with NDFC0_CR[CER]='0' and the CE# signal would otherwise stay asserted.

1	ERC	ECC Reset 0 no action 1 Reset ECC	Setting this bit to 1 resets the ECC counter(s)/calculation(s). Note: Writes to this bit are not actually stored in the register, but rather only serve to perform the reset function.
2	RIE	Interrupt Enable on Device Ready 0 No Interrupt output 1 Interrupt pulse generated when RDY/BSY# goes from 0 -> 1, indicating device has just completed some internal operation.	The Interrupt pulse will be generated on the NDFC_IRQ core output and will be 4 PerClk wide. Typically this means that the interrupt controller must detect this signal with an edge-sensitive input.
3	REN	Enable wait for Ready in reads from Linear Region 0 NDFC ignores value of NFRdyBusy when accessing the linear data region 1 NDFC waits for NFRdyBusy == 1 (ready) before allowing read cycles to occur in the linear data region.	Setting this bit also allows the NDFC0_BxCR[RR] timer to be activated when a read is performed in the linear region and the current, latched state of NFRdyBusy== 0 (busy). The NDFC0_BxCR[RR] counter counts when NFRdyBusy goes from 0 (busy) to 1 (ready) before allowing a read cycle to the external device. Clearing this bit means that software must have insured that the device is ready prior to accessing the external device. Clearing this bit is required since the NFRdyBusy signal is multiplexed with other signals. This bit can be strapped at reset time via SDR0_SDSTP2[NRB]. Note: This setting also affects AutoRead mode because accesses at reset time for AutoRead mode are sent to the linear access region.
4		Reserved	
5	ARE	Auto-Read Enable 0 Reads to Linear Region only generate read cycle to the external NAND Flash. 1 Reads to Linear Region can generate automatic page read commands with the address determined from the EBC read address and then perform a read cycle to the external NAND flash.	When set, the address of reads to the Linear region will be compared to the previous read address and if the new address is 'non-sequential' (i.e., it is not <prev_addr>+4), then an automatic 'page read' command will be performed. The address of the page read command will be determined from the new address following the rules outlined in *AutoRead Mode* on page 589.
6:7	BS	Bank Select 00 Select Bank on CE[0] 01 Select Bank on CE[1] 10 Select Bank on CE[2] 11 Select Bank on CE[3]	Changing the Setting from 0 to 3; setting determines which bank is 'selected' for subsequent hardware accesses. Changing the Bank Select setting will also deassert any active chip enable (the same as writing NDFC0_CR[CER]=1).
8:17		Reserved	
18:19	ARAC	Auto-Read mode Address Cycles 00 3 Addr. Cycles, 1 Col. + 2 Row (512 page size) 01 4 Addr. Cycles, 1 Col. + 3 Row (512 page size) 10 4 Addr. Cycles, 2 Col. + 2 Row (2k page size) 11 5 Addr. Cycles, 2 Col. + 3 Row (2k page size)	Used when AutoRead mode is enabled. This value determines how many address cycles are performed during the setup to configure the external device for reading the IPL from bank 0. This value is preloaded from the SDR0_SDSTP2[NBAC] strapping input during reset. See *AutoRead Mode* on page 589.

20:23	RPG	Auto-Read mode base Page value	Used when AutoRead mode is enabled. This value determines the base page address that an automatically generated page read command will output to the NAND Flash during AutoRead. This value is preloaded from the SDR0_SDSTP2[NBP] strapping input during reset. If this value is changed from it's strapped, default value, the software must take care to insure that the next AutoRead mode access made to the NDFC linear region is non-sequential so that the new page address value will be sent to the external NAND Flash device. This is because writing to the NDFC0_CR register by itself does not cause the NDFC to assume that the ARPG value has changed. See *AutoRead Mode* on page 589.
24:28		Reserved	
29	EBCC	EBC Configuration Completed 0 Gate clocks to the EBC to pace transactions when in AutoRead mode. Only used when booting from NAND flash. 1 Never gate clocks to the EBC. Required setting for normal operation.	Immediately after reset, the EBC PerReady signal is not enable. To pace transactions when booting from NAND flash in AutoRead mode, the NDFC gates the clocks to the EBC until the transactions complete on the NDFC interface. EBCC must be set to 1 for normal operation.
30	DHC	Direct Hardware Control Enable 0 Normal 1 Use NDFC0_HWCTL register to directly control and read external interface inputs and outputs	This bit enables the NDFC0_HWCTL register to control the NDFC external interface directly. The automatic control from the NDFC0_CMD, NDFC0_ADDR, NDFC0_DATA registers and the linear address region(s) are disabled when DHC is set. This should be used for debug purposes only.
31		Reserved	

23.9.8 NDFC Status Register (NDFC0_SR)

The NAND Flash Status Register (NDFC0_SR) is for monitoring the external Ready/Busy# indicator input that comes from the NAND Flash devices, indicating the currently active ECC accumulation register, and a status bit to indicate that read data loss may have occurred in accesses to an external NAND Flash device.

This register cannot be written. Unused bits return a value of '0'.

Figure 23-18. NAND Flash Status Register (NDFC0_SR)

0		Reserved	
1:3	ECCP	Active NDFC0_ECC register identifier 0-7 = NDFC0_ECC0:NDFC0_ECC7	This field is reset to '000' when the NDFC0_CR[ERC] bit is written with a '1' to generate an ECC Reset. This field is automatically incremented after every 256-Bytes of data are transferred to the external interface. Once ECCP reaches '111' (NDFC0_ECC7) and 256 bytes of data are transferred, the field will not be incremented any further. This allows a full 2kB page to be transferred followed by its 64B spare data area without the spare data affecting the ECC value stored in any of the NDFC0_ECC registers.
4:5		Reserved	

6	RDL	Read Data Loss 0 No Read Data Loss has been detected 1 Possible Read data has been lost	This bit is only set when a EBC single byte-wide read access is performed to an external NAND Flash bank that is configured to be 16-bit wide, indicating that the upper 8-bits of read data from that device access were not returned. This may occur if byte read cycles are performed by the EBC to a 16-bit NAND flash bank. This bit is only cleared when an CE# Reset via NDFC0_CR[CER] is performed. Note: This status bit is not cleared by changing the NDFC0_CR[BS] bank select.
7	RDY	Ready/Busy# Input indicator 0 External NFRdyBusy# input is low -> Busy 1 ExternalNFRdyBusy# input is high -> Ready	
8:31		Reserved	

23.9.9 NAND Flash Direct Hardware Control Register (NDFC0_HWCTL)

The NAND Flash Direct Hardware Control Register (NDFC0_HWCTL) enables direct control of the external interface inputs and outputs that go to the NAND Flash devices. The settings in this register have no control over the external interface unless the NDFC0_CR[DHC] bit is set.

This register is Read/Write with several conditions.

1. The contents of this register are not writable and reads will not return data in the DATAL and DATAH fields unless the NDFC0_CR[DHC] bit is already set.
2. The DATAL and DATAH fields are read-only when the DENL and DENH fields respectively are cleared (=0).

Unused bits return a value of 0

Figure 23-19. NAND Flash Direct Hardware Control Register (NDFC0_HWCTL)

0:3	CE0-CE3	Chip Enable $\overline{\text{NFCE0:3}}$	This 4-bit field contains bits CE0:CE3. The setting of these bits is inverted before going to the external output, so a setting of '1' here gives an active output (low) to the NAND Flash device's CE# input.
4	CLE	$\overline{\text{NFCLE}}$ direct control 0 $\overline{\text{NFCLE}}$ = 0 1 $\overline{\text{NFCLE}}$ = 1	The value of this bit directly correlates to the external output signal
5	ALE	NFALE direct control 0 $\overline{\text{NFALE}}$ = 0 1 $\overline{\text{NFALE}}$ = 1	The value of this bit directly correlates to the external output signal
6	RE	$\overline{\text{NFREn}}$ direct control (inverted) 0 $\overline{\text{NFREn}}$ = 1 1 $\overline{\text{NFREn}}$ = 0	The value of this bit is inverted before driving the external output signal.
7	WE	$\overline{\text{NFWEn}}$ direct control (inverted) 0 $\overline{\text{NFWEn}}$ = 1 1 $\overline{\text{NFWEn}}$ = 0	The value of this bit is inverted before driving the external output signal.
8	DENL	Direct control of External Device Write Data bus enable for NFData 00:07 0 PerData00:07 drivers are High-Z (Read) 1 PerData00:07 drivers are not High-Z (Write)	
9	DENH	0 Direct control of External Device Write Data bus enable for NFDATA08:15 0 PerData08:15 drivers are High-Z (Read) 1 PerData08:15 drivers are not High-Z (Write)	

10:15		Reserved	
16:23	DATAL	PerData00:07 direct control (DENL = 1) Read/Write PerData00:07 latched input value (DENL = 0) Read-only	When DENL is 0, this field returns the value on the PerData00:07 inputs. When DENL is 1, this field drives its value out to the PerData00:07 outputs.
24:31	DATAH	PerData08:15 direct control (DENH=1) Read/Write PerData08:15 latched input value (DENH = 0) Read-only	When DENH is 0, this field returns the value on the PerData08:15 inputs. When DENH is 1, this field drives its value out to the PerData08:15 outputs.

23.9.10 NDFC Revision ID Register (NDFC0_REVID)

NDFC0_REVID is a read-only register that allows software to read the version information.

Figure 23-20. NAND Flash Revision ID Register (NDFC0_REVID)

0:11		Reserved	
12:23	RN	Revision Number	Corresponds to the RCS major revision number of the source RTL
24:31	BRN	Branch Revision Number	Corresponds to the RCS minor (branch) revision number of the source RTL

24. Direct Memory Access Controllers

PPC440EPx/GRx implements two direct memory access controllers: one attached to 128-bit processor local bus (DMA2P40) and the other attached to 64-bit processor local bus (DMA2P30). Because of functional differences this chapter is organized under two separate sections as follows:

- *DMA to PLB4 Controller (DMA2P40)*
- *DMA to PLB3 Controller (DMA2P30)*

The Direct Memory Access (DMA) controller is a Processor Local Bus (PLB) and On-chip Peripheral Bus (OPB) master which supports the autonomous transfer of data between memory and peripherals and from memory-to-memory. The controller provides four DMA channels, each of which has an independent set of configuration registers. Each channel has its own control, count and control, source address, destination address, and scatter/gather address registers. Once these registers are programmed by the PPC440EPx/GRx processor, the DMA controller performs the requested data transfer without the need for processor intervention.

The four DMA channels also support scatter/gather transfers. During a scatter/gather transfer, the configuration registers for a particular DMA channel are automatically loaded from a data structure in memory instead of being individually programmed. Since the scatter/gather address register is updated in this process, the channel can optionally reconfigure itself for another transfer when the current one completes.

As PLB and OPB master, the DMA can read and write any address accessible by the PPC440EPx/GRx processor. This includes memory and memory-mapped peripherals on the external bus controller (EBC) interface and SDRAM memory and PCI addresses that have been mapped into PLB address space. The DMA controller can also service DMA peripherals attached to the EBC via the DMAReqn, DMAAckn and EOTn[TCn] I/Os, along with the OPB-attached UART0 and UART1.

24.1 DMA to PLB4 Controller (DMA2P40)

DMA to PLB4 is attached to the 128-bit processor local bus. Neither sub-channels nor external sources are used by this controller.

Table 24-1 lists the DMA to PLB4 channel assignments in the PPC440EPx/GRx.

Table 24-1. DMA to PLB 4 Channel Assignments

Channel	DMA Internal Source
Channel 0	USB2.0 Device Endpoint 1 In *(PPC440EPx only)*
Channel 1	USB2.0 Device Endpoint 2 In *(PPC440EPx only)*
Channel 2	USB2.0 Device Endpoint 1Out *(PPC440EPx only)*
Channel 3	USB2.0 Device Endpoint 2 Out *(PPC440EPx only)*

24.1.1 DMA Transfers

As a specialized controller, the DMA provides system designers and programmers with a highly efficient method of moving data. During any DMA transfer the controller always buffers data read from the source prior to writing the data to the destination. Since many buses, including the internal PLB, provide substantially better performance

when bursting data, the DMA controller includes a 128-byte (8 quad word) buffer. This buffer is enabled on a per-channel basis by setting DMA2P40_CRn[BEN] and serves to minimize the number of discrete memory transactions. Each of the four DMA channels is configurable for either peripheral or memory-to-memory transfers.

24.1.1.1 Memory-to-Memory Transfers

The DMA controller can perform either device-paced (hardware-initiated) or software-initiated memory-to-memory transfers

Device-Paced Memory Mode Transfers (Hardware Initiated)

Device-paced memory (DPM) corresponds to the USB 2.0 device channels listed in *Table 24-1*. Memory is any address accessible from the PLB or OPB, including PLB-mapped PCI address space, SDRAM memory, and memory connected to the external bus.

During a DPM mode transfer, the DPM device requests a DMA transfer by asserting a DMA request line. For internal DPMs, this signal is internal to the chip. When the requesting channel has the highest priority of any active channel, the DMA core executes a DPM transfer.

There are two types of DPM transfers: DPM-to-memory and memory-to-DPM. A DPM-to-memory transfer reads data from a DMA DPM device, while a memory-to-DPM transfer writes data. In both cases, the DPM interface transfers data at the programmed width of the peripheral (DMA2P40_CRn[PW]). When the DMA buffer is disabled for the active channel (DMA2P40_CRn[BEN]=0), each DPM transfer causes a corresponding memory operation. When buffering is enabled during a DPM-to-memory transfer, data is collected until the 128-byte buffer is full, the DPM deasserts DMAReqn, or the channel completes. The buffer contents are then written to the target memory as efficiently as possible.

If the Destination address is located on PLB space, the data is written to memory with one burst operation if the destination address is quad word aligned and the buffer is full; or, it is written with a combination of single beat and burst transfers, depending on the destination address alignment and the amount of data stored in the DMA buffer. If the destination address is located on OPB memory space, the DMA controller writes out the content of the DMA buffer with a series of single beat or burst transfers, depending on whether bursting is enabled.

Memory-to-DPM transfers differ since the amount of data that will be requested by the DPM is unknown. If the DMA buffer is disabled (DMA2P40_CRn[BEN]=0), a discrete source memory read occurs for each element in the DMA transfer. Since this is inefficient, the buffer should only be disabled for low data rate transfers. When the 128-byte buffer is enabled, the controller uses the setting in DMA2P40_CRn[PF] to prefetch 1, 2, 4, or 8 128-bit quad words from the source memory. The DMA controller provides data from the buffer until the DPM deasserts its request or the transfer completes. Whenever any of these conditions occurs, any unused data in the DMA buffer is discarded.

For both DPM-to-memory and memory-to-DPM transfers, the DMA controller supports fixed length bursts of 2, 4, 8, or 16 data elements on the peripheral side of the transfer. When bursting is enabled (DMA2P40_CTn[BTEN]=1), an active DMA request (DMAReqn) causes the DMA controller to transfer the programmed number of data items (DMA2P40_CTn[BSIZ]) in one atomic operation.

Software-Initiated Memory-to-Memory Transfers

Software-initiated memory-to-memory transfers between memories with fixed timings provide the best overall performance. During a software-initiated transfer, the DMA controller knows the exact amount of data to be transferred. As a result, when the 128-byte DMA buffer is enabled (DMA2P40_CRn[BEN]=1) the controller uses bursts as much as possible if the memory is located on PLB space. For memory located on OPB space, bursting has to be enabled (DMA2P40_CTn[BTEN]=1) for the DMA controller to burst. To ensure the highest bandwidth with memory locate on PLB space, source and destination addresses should be aligned on 128-byte boundaries.

There are two special cases of software-initiated memory-to-memory transfers.

- Source address increment bit is reset (DMA2P40_CRn[SAI]=0).

  In this special case, the source memory device responds to a unique address. If the source memory device (FIFO-like device) is located on PLB space, bursting is not possible. If the source memory device is located on OPB space, bursting is possible if enabled (DMA2P40_CTn[BTEN]=1).

- Destination address increment bit is reset (DMA2P40_CRn[DAI]=0).

  In this case, the destination memory device responds to a unique address. If the destination memory device (FIFO-like device) is located on PLB space, bursting is not possible. If the destination memory device is located on OPB space, bursting is possible if enabled (DMA2P40_CTn[BTEN]=1).

24.1.1.2 Scatter/Gather Transfers

Each of the four DMA channels supports scatter/gather transfers. This scatter/gather capability allows the chaining of multiple DMA controller operations within a channel. During a normal DMA operation software must program the control, count and control, source address, and destination address registers for each transfer. Scatter/gather transfers differ in that these registers are automatically loaded from a linked list data structure in system memory. When a channel completes one transfer, the DMA controller loads the next set of configuration values into the channel registers and the channel continues with the new programming.

24.1.2 Configuration and Status Registers

Table 24-2 lists the DMA configuration and status registers, which are accessed using the **mtdcr** and **mfdcr** instructions. As example, the following PowerPC assembly code writes the control register for DMA channel 0 and then reads the DMA status register:

```
#define DMA2P40_CR0  0x100
#define DMA2P40_SR   0x120

    mtdcr     DMA2P40_CR0,r3            ! write r3 to channel 0 control register
    mfdcr     r4,DMA2P40_SR             ! read contents of status register into r4
```

The DMA configuration and status registers are readable at any time; however, since each register read requires a separate operation, it is not possible to guarantee that the values read from multiple registers correspond to a state that ever existed in the DMA controller. To illustrate, consider software that reads the destination address registers for channel 0 (DMA2P40_DAH0 and DMA2P40_DAL0) and the count for channel 0 (DMA2P40_CTC0). If the DMA controller updates the destination address or count between these operations, the values read differ from what is expected.

While reads can occur at any time, software must not write the configuration registers for any channel that is currently enabled (DMA2P40_CRn[CE]=1). The only exception is that a channel may be disabled by reading the channel control register, clearing the channel enable bit, and then writing the new value to the control register. Note that before enabling the channel enable bit, the status has to be queried until the channel busy status bit is cleared. Once a channel is disabled, all of its configuration registers may be reprogrammed as desired.

Table 24-2. DMA to PLB4 Controller Configuration and Status Registers

Mnemonic	Name	DCR Number	Access	Page
DMA2P40_CR0	DMA to PLB4 Channel Control Register 0	0x0300	R/W	613
DMA2P40_CTC0	DMA to PLB 4 Count and Control Register 0	0x0301	R/W	614
DMA2P40_SAH0	DMA to PLB 4 Source Address High Register 0	0x0302	R/W	615
DMA2P40_SAL0	DMA to PLB 4 Source Address Low Register 0	0x0303	R/W	615

Table 24-2. DMA to PLB4 Controller Configuration and Status Registers (continued)

Mnemonic	Name	DCR Number	Access	Page
DMA2P40_DAH0	DMA to PLB 4 Destination Address High Register 0	0x0304	R/W	616
DMA2P40_DAL0	DMA to PLB 4 Destination Address Low Register 0	0x0305	R/W	616
DMA2P40_SGH0	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 0	0x0306	R/W	616
DMA2P40_SGL0	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 0	0x0307	R/W	616
DMA2P40_CR1	DMA to PLB 4 Channel Control Register 1	0x0308	R/W	613
DMA2P40_CTC1	DMA to PLB 4 Count and Control Register 1	0x0309	R/W	614
DMA2P40_SAH1	DMA to PLB 4 Source Address High Register 1	0x030A	R/W	615
DMA2P40_SAL1	DMA to PLB 4 Source Address Low Register 1	0x030B	R/W	615
DMA2P40_DAH1	DMA to PLB 4 Destination Address High Register 1	0x030C	R/W	616
DMA2P40_DAL1	DMA to PLB 4 Destination Address Low Register 1	0x030D	R/W	616
DMA2P40_SGH1	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 1	0x030E	R/W	616
DMA2P40_SGL1	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 1	0x030F	R/W	616
DMA2P40_CR2	DMA to PLB 4 Channel Control Register 2	0x0310	R/W	613
DMA2P40_CTC2	DMA to PLB 4 Count and Control Register 2	0x0311	R/W	614
DMA2P40_SAH2	DMA to PLB 4 Source Address High Register 2	0x0312	R/W	615
DMA2P40_SAL2	DMA to PLB 4 Source Address Low Register 2	0x0313	R/W	615
DMA2P40_DAH2	DMA to PLB 4 Destination Address High Register 2	0x0314	R/W	616
DMA2P40_DAL2	DMA to PLB 4 Destination Address Low Register 2	0x0315	R/W	616
DMA2P40_SGH2	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 2	0x0316	R/W	616
DMA2P40_SGL2	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 2	0x0317	R/W	616
DMA2P40_CR3	DMA to PLB 4 Channel Control Register 3	0x0318	R/W	613
DMA2P40_CTC3	DMA to PLB 4 Count and Control Register 3	0x0319	R/W	614
DMA2P40_SAH3	DMA to PLB 4 Source Address High Register 3	0x031A	R/W	615
DMA2P40_SAL3	DMA to PLB 4 Source Address Low Register 3	0x031B	R/W	615
DMA2P40_DAH3	DMA to PLB 4 Destination Address High Register 3	0x031C	R/W	616
DMA2P40_DAL3	DMA to PLB 4 Destination Address Low Register 3	0x031D	R/W	616
DMA2P40_SGH3	DMA to PLB 4 Scatter/Gather Descriptor Address High Register 3	0x031E	R/W	616
DMA2P40_SGL3	DMA to PLB 4 Scatter/Gather Descriptor Address Low Register 3	0x031F	R/W	616
DMA2P40_SR	DMA to PLB 4 Status Register	0x0320	R/Clear	617
DMA2P40_SGC	DMA to PLB 4 Scatter/Gather Command Register	0x0323	R/W	618
DMA2P40_SLP	DMA to PLB 4 Sleep Mode Register	0x0325	R/W	618
DMA2P40_POL	DMA to PLB 4 Polarity Configuration Register	0x0326	R/W	619

24.1.2.1 DMA to PLB 4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3)

DMA channel control registers (DMA2P40_CR0-DMA2P40_CR3) are used to configure and enable their respective DMA channels. Before a DMA channel can transfer data, the channel control, count and control, source address and destination address registers must be programmed. If a DMA channel is set up for scatter/gather transfers (DMA2P40_SGC[SSGn]=1), the DMA channel control register is automatically loaded from memory (see *Scatter/Gather Transfers* on page 622).

Figure 24-1. DMA to PLB4 Channel Control Registers (DMA2P40_CR0-DMA2P40_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA2P40_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Not allowed 1 Device located on the OPB.	For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers, the device is always located on the OPB.
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Half word (16 bits) 010 Word (32 bits) 011 Double word (64 bits) 100 Quad word (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, half word, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quad word size. Use 010 (32 bits) for USB 2.0 Device controller.
7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: • 1, for byte (8-bit) transfers • 2, for half word (16-bit) transfers • 4, for word (32-bit) transfers • 8, for double word (64-bit) transfers • 16, for quad word (128-bit) transfers	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.
8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: • 1, for byte (8-bit) transfers • 2, for half word (16-bit) transfers • 4, for word (32-bit) transfers • 8, for double word (64-bit) transfers • 16, for quad word (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.

9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Reserved 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	Must specify 11 for USB.
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers. Select 0 cycles for USB.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers. Select 0 cycles for USB.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers. Select 0 cycles for USB.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quad word (128-bits) 01 Prefetch 2 quad words 10 Prefetch 4 quad words 11 Prefetch 8 quad words	Used only during memory-to-peripheral and memory to device-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 Reserved	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller. Must be 0 for USB.
31		Reserved	

24.1.2.2 DMA to PLB 4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3))

The DMA count and control registers (DMA2P40_CT0-DMA2P40_CT3) contain the number of transfers remaining in the DMA transaction for their respective channels when EOTn[TCn] is programmed as a terminal count output, along with additional channel control information. In addition to the transfer count, the count and control registers contain fields used to enable interrupts for terminal count, end of transfer and error conditions. Other fields enable

parity checking during the peripheral portion of peripheral type (DMA2P40_CRn[TM]=00) transfers, define sub channel ID information, and enable bursting during peripheral and device-paced memory transfers. For additional details on bursting, see *Peripheral and Device-Paced Memory Bursts* on page 619.

When a DMA channel is setup for scatter/gather transfers (DMA2P40_SGC[SSGn]=1) the count and control register is automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 622.

The value in the Transfer Count (TC) field of DMA count and control register is interpreted as the number of transfers of the width specified in DMA2P40_CRn[PW], not the total number of bytes. The maximum number of transfers is 1024K, and each transfer can be either 1, 2, 4, 8, or 16 bytes as programmed in DMA2P40_CRn[PW]. The maximum count of 1024K transfers is programmed by writing zero to DMA2P40_CTn[TC].

The sub channel ID bits can be used to allow one DMA channel to support up to eight different peripherals. Bits 5:7 control the external logic that selects the current peripheral.

When EOTn[TCn] is programmed as an end-of-transfer input (DMA2P40_CRn[ETD]=0), the channel will not stop when the count reaches zero. Instead, DMA2P40_CTn[TC] continues to count down past zero until EOTn is asserted. For DMA2P40_CRn[ETD]=0, DMA2P40_CTn[TCE] must be set to zero.

Figure 24-2. DMA to PLB4 Count and Control Registers (DMA2P40_CT0-DMA2P40_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA2P40_CR0-DMA2P40_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Disable end of transfer interrupts 1 Enable end of transfer interrupts	EOT interrupts are further qualified by DMA2P40_CR0-DMA2P40_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Disable error interrupts 1 Enable error interrupts	Error interrupts are further qualified by DMA2P40_CR0-DMA2P40_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Device-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRSTSIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

24.1.2.3 DMA to PLB 4 Source Address Registers

The DMA Source Address Registers (DMA2P40_SAHn and DMA2P40_SALn) contain the source address for memory-to-memory and memory-to-peripheral transfers. If a DMA channel is setup for scatter/gather transfers (DMA2P40_SGC[SSGn]=1) the source address registers are automatically loaded from memory. For additional details *Scatter/Gather Transfers* on page 622

The source address must be aligned at the transfer width programmed in DMA2P40_CRn[TW], otherwise the error bit (DMA2P40_SR[RIn]) is set for the channel and no transfer occurs. If the source address increment bit in the channel's control register is set (DMA2P40_CRn[SAI]) the address is incremented by the transfer width after each data transfer. In contrast, if the channel is performing a memory-to-peripheral transfer and the address decrement bit is set (DMA2P40_CRn[DEC]=1), the address is decremented by the transfer width after each transfer.

Figure 24-3. DMA to PLB4 Source Address High Registers (DMA2P40-SAH0-DMA2P40-SAH3)

0:27		Reserved	
28:31	USA	Upper 4-bits of source address for memory-to-memory and memory-to-peripheral transfers.	

Figure 24-4. DMA to PLB4 Source Address Low Registers (DMA2P40-SAL0-DMA2P40-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.	

24.1.2.4 DMA to PLB 4 Destination Address Registers

DMA destination address registers (DMA2P40_DAH0-DMA2P40_DAH3 and DMA2P40_DAL0-DMA2P40_DAL3) contain the destination address for memory-to-memory and peripheral-to-memory transfers. When a DMA channel is configured for scatter/gather transfers (DMA_SGC[SSGn]=1) the destination address registers are automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 622

The destination address must be aligned at the transfer width programmed in DMA2P40_CRn[TW], otherwise the error bit (DMA2P40_SR[RIn]) is set for the channel and no transfer occurs. If the destination address increment bit in the channel's control register is set (DMA2P40_CRn[DAI]) the address is incremented by the transfer width after each data transfer. However, if the channel is performing a peripheral-to-memory transfer and the address decrement bit is set (DMA2P40_CRn[DEC]=1), the destination address is decremented by the transfer width after each transfer.

Figure 24-5. DMA to PLB4 Destination Address High Registers (DMA2P40_DAH0-DMA2P40_DAH3)

0:27		Reserved	
28:31	UDA	Upper 4-bits of destination address for memory-to-memory and peripheral-to-memory transfers.	

Figure 24-6. DMA to PLB4 Destination Address Low Registers (DMA2P40_DAL0-DMA2P40_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.	

24.1.2.5 DMA to PLB 4 Scatter/Gather Descriptor Address Registers

When a DMA channel is setup for scatter/gather transfers (DMA2P40_SGC[SSGn]=1), the Scatter/Gather Descriptor Address Registers (DMA2P40_SGH0-DMA2P40_SGH3 and DMA2P40_SGL0-DMA2P40_SGL3) contain the memory address of the next scatter/gather descriptor table. Prior to starting a scatter/gather transfer,

software must write the address of the channel's descriptor table to DMA2P40_SGHn and DMA2P40_SGLn. Once the scatter/gather transfer starts, DMA2P40_SGHn and DMA2P40_SGLn are automatically updated from the descriptor table. For additional details see *Scatter/Gather Transfers* on page 622.

Figure 24-7. DMA to PLB4 Scatter/Gather Desc Adr High Reg (DMA2P40_SGH0-DMA2P40_SGH3)

0:27		Reserved	
28:31	USGD	Upper 4-bits of address of next scatter/gather descriptor table.	

Figure 24-8. DMA to PLB4 Scatter/Gather Desc Adr Low Reg (DMA2P40_SGL0-DMA2P40_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.	

24.1.2.6 DMA to PLB 4 Status Register (DMA2P40_SR)

DMA Status Register (DMA2P40_SR) provides status information for each of the DMA channels. Bits in DMA2P40_SR are set in hardware, and can be either read or cleared by software. Clearing is performed by writing a word to DMA2P40_SR containing a 1 in any bit position to be cleared and 0 in all other bit positions.

The terminal count status (DMA2P40_SR[CSn]), end of transfer status (DMA2P40_SR[TSn]) and error status (DMA2P40_SR[RIn]) must be cleared for a DMA channel to operate. If a scatter/gather operation generates an interrupt for any of the above conditions, the channel pauses until software clears the associated status field(s) in DMA2P40_SR.

Figure 24-9. DMA to PLB4 Status Register (DMA2P40_SR)

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA2P40_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA2P40_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During scatter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept active until the last link is completed.
28:31		Reserved	

24.1.2.7 DMA to PLB 4 Scatter/Gather Command Register (DMA2P40_SGC)

The DMA Scatter/Gather Command Register (DMA2P40_SGC) is a 32-bit register, of which 12-bits are implemented. Bits 0:3 are the Start Scatter/Gather Enable bits for channels 0 to 3, bits 4:7 determine whether a given channel's Scatter/Gather Descriptor Table resides in PLB or OPB address space, and bits 16:19 are the corresponding Enable Mask bits for the Start Scatter/Gather Enable bits. Setting a Start Scatter/Gather Enable bit causes the selected channel to begin a scatter/gather operation, while writing a 0 stops the Scatter/Gather operation. To start or stop a specific Scatter/Gather channel, the corresponding Enable Mask bit must be set to 1; otherwise, the register holds the previous value. Note that halting a scatter/gather transfer prevents the channel from continuing to the next descriptor, but does not stop the transfer currently in progress.

Upon completion of a scatter/gather sequence of transfers the DMA controller clears DMA2P40_SGC[SSGn]. If an error occurs when the DMA controller is reading the Scatter/Gather descriptor table, DMA2P40_SGC[SSGn] is cleared for the affected channel, and the channel's error status bit (DMA2P40_SR[RIn]) is set. For additional details see *Scatter/Gather Transfers* on page 622.

Figure 24-10. DMA to PLB4 Scatter/Gather Command Register (DMA2P40_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See *Scatter/Gather Transfers* on page 622.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

24.1.2.8 DMA to PLB 4 Sleep Mode Register (DMA2P40_SLP)

The Sleep Mode Register (DMA2P40_SLP) enables the DMA controller to enter sleep (low-power) mode and programs the number of PLB clock cycles to wait when the controller is idle before going to sleep. The DMA controller only goes to sleep when no DMA channels are enabled, no scatter/gather operation is pending, and no configuration or status (DCR) register operations are in progress. Reading or writing any of the DMA DCRs awakens the DMA controller.

To enable sleep mode set DMA2P40_SLP[SME] and CPM0_ER[DMA]. When sleep mode is enabled and the DMA controller becomes idle the 10-bit idle timer begins counting down from the programmed value. Only the upper 5 bits of the idle counter are programmable; the lower 5 bits are hard coded to 5'0b11111; therefore, the minimum granularity of the idle timer is 32 PLB clock cycles. When the counter reaches zero, the controller is placed in sleep mode.

Figure 24-11. DMA to PLB4 Sleep Mode Register (DMA2P40_SLP)			
0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hard coded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

24.1.2.9 DMA to PLB 4 Polarity Configuration Register (DMA2P40_POL)

The Polarity Configuration Register (DMA2P40_POL) is present but is reserved since external DMA is not supported.

24.1.3 Channel Priorities

The priority of DMA transfers is controlled on a per-channel basis by the channel priority field in the channel control register. *Table 24-3* shows the different priority settings for DMA2P40_CRn[PW].

Table 24-3. DMA to PLB 4 Transfer Priorities

DMA2P40_CRx[CP]	Priority Level
0b00	Low
0b01	Medium Low
0b10	Medium High
0b11	High

These priorities serve two purposes. First, the DMA controller arbitrates among all actively requesting channels and selects the highest priority one for service. If multiple channels request at the same priority, the arbiter selects the lowest numbered channel for service. Secondly, DMA2P40_CRn[CP] determines the priority of the internal PLB transactions that the DMA controller uses to read and write data. During a Scatter/Gather transfer, the next descriptor is read using the priority current programmed in DMA2P40_CRn[CP].

24.1.4 Data Parity During DMA Peripheral Transfers

During memory-to-memory transfers any data error checking and/or correction is dependent on the configuration of the relevant memory controller. For example, if ECC is enabled on the SDRAM controller, only uncorrectable errors are reported to the DMA controller.

24.1.5 Peripheral and Device-Paced Memory Bursts

Normally, device-paced (hardware initiated) DMA transfers move one data item at a time.

To improve performance, the DMA controller supports fixed-length bursts on the device-paced side of device-paced memory-to-memory transfers. When bursting is enabled (DMA2P40_CTn[BTEN]=1), the DMA controller responds to an active DMAReqn by reading or writing DMA2P40_CTn[BSIZ] data items in one atomic, non-interruptible operation.

When a DMA channel has bursting enabled (DMA2P40_CTn[BTEN]=1) it is required that the prefetch count (DMA2P40_CRn[PF]) be sufficient to complete at least one fixed-length burst transfer. In addition, the transfer count (DMA2P40_CTn[TC]) must be a multiple of the burst size (DMA2P40_CTn[BSIZ]). If either of these requirements is not met, an error results, no data transfer occurs, and the channel is disabled. For additional details see *Errors* on page 620

24.1.6 Errors

The DMA controller detects and reports five types of errors: address alignment, burst count, burst prefetch, PLB/OPB time-out, and slave errors. The DMA controller reports errors through the channel error status bit in the DMA status register (DMA2P40_SR[RIn]). Whenever the error status bit for a channel is set, the channel enable bit (DMA2P40_CRn[CE]) is cleared, disabling the channel. An interrupt signal is also presented to the interrupt controller if DMA2P40_CRn[CIE] and DMA2P40_CTn[EIE] are set. For more information on interrupt processing, see *DMA to PLB4 Interrupts* on page 621

When an error is detected during the execution of an external cycle, the DMA terminates the transfer and deasserts all external signals.

When the DMA controller has multiple channels active, an error may be reported on the current channel which was in actuality caused by a previously active channel. This causes the current channel to have its error status bit set; therefore, for deterministic error analysis with multiple DMA channels active, the slave bus controller's error status registers (the bus error address register in particular) must be queried to isolate the actual channel that encountered the error. In any case, the channel causing the errors will eventually cause all active channels, including itself, to be disabled.

24.1.6.1 Address Alignment Error

The source address (DMA2P40_SAHn || DMA2P40_SALn) and destination address (DMA2P40_DAHn || DMA2P40_DALn) registers must be aligned to the programmed transfer width (DMA2P40_CRn[PW]).The address alignment rules are outlined in *Table 24-4*. In addition, when a channel is configured for scatter/gather transfers, the scatter/gather table must be quad word-aligned. If the source, destination, and scatter/gather address registers are not appropriately aligned, an error occurs immediately after the channel is enabled.

Table 24-4. Address Alignment Requirements

DMA2P40_CRx[PW] Setting	Required Alignment for: Source Address (DMA2P40_SAHn || DMA2P40_SALn) Destination Address (DMA2P40_DAHn || DMA2P40_DALn)
0b000	Byte (8-bit)
0b001	Halfword (16-bit)
0b010	Word (32-bit)
0b011	Doubleword (64-bit)
0b100	Quadword (128-bit)

24.1.6.2 Burst Count Error

If peripheral/device-paced bursting is enabled for a particular DMA channel (DMA2P40_CTn[BTEN]=1), the programmed transfer count (DMA2P40_CTn[TC]) must be a multiple of the burst size (DMA2P40_CTn[BSIZ]).

24.1.6.3 Burst Prefetch Error

If peripheral/device-paced bursting is enabled for a particular DMA channel (DMA2P40_CTn[BTEN]=1), the programmed prefetch count (DMA2P40_CRn[PF]) must be sufficient to perform at least one fixed-length burst. More precisely, DMA2P40_CRn[PF] must prefetch at least DMA2P40_CTn[BSIZ] * DMA2P40_CRn[PW] bytes.

24.1.6.4 PLB or OPB Timeout

The DMA controller uses PLB and OPB operations to read and write memory. A PLB or OPB timeout results if the DMA controller attempts to access a non-existent memory location. This will occur if the source, destination or scatter/gather address registers do not map to valid memory locations.

24.1.6.5 Slave Transfer Errors

If the DMA controller detects an error from a slave, it stops the execution of the current transfer, disables the channel, and then reports an error. If a signal is asserted by the PLB slave as a result of such error, the DMA finishes any active read/write pair transfer before disabling the channel. An SDRAM uncorrectable ECC error and an EBC bank protection error are examples of slave errors.

24.1.7 DMA to PLB4 Interrupts

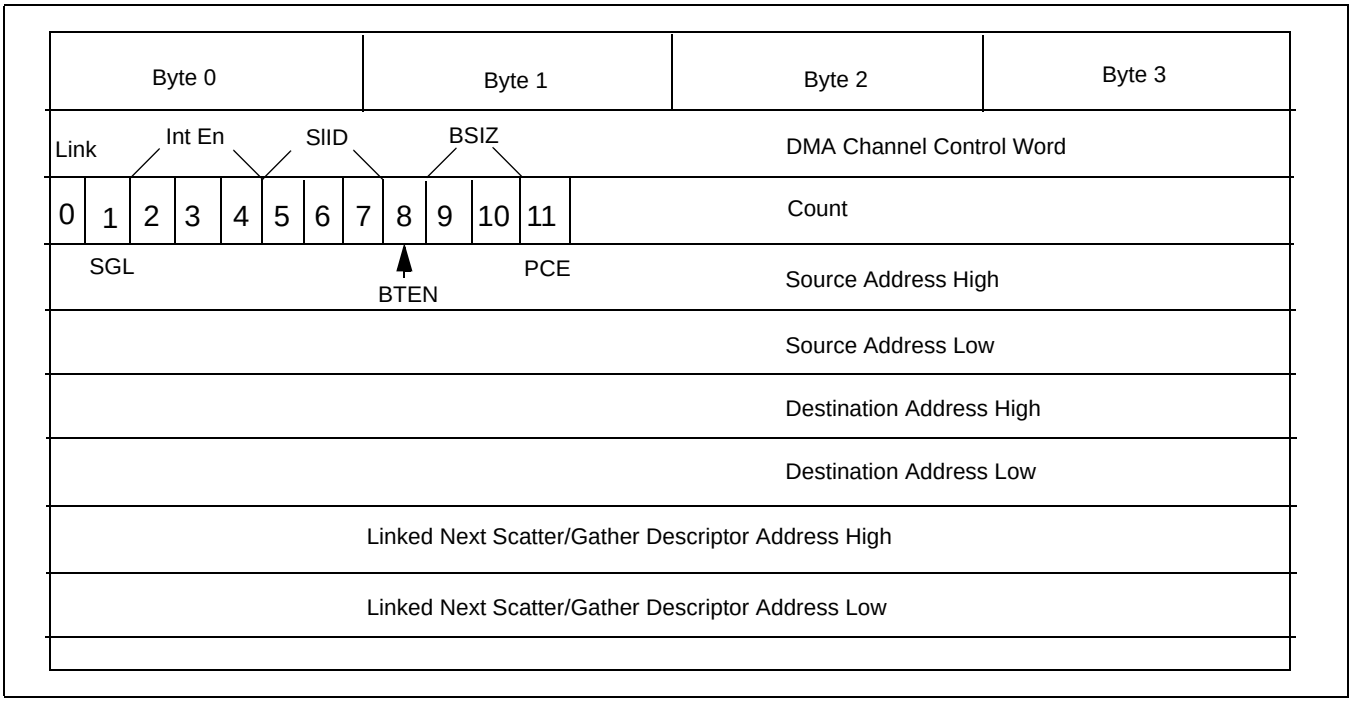
Each DMA channel can generate interrupts for end of transfer, terminal count and error conditions. All interrupts from a particular DMA channel are enabled by setting the channel enable bit in the channel's control register (DMA2P40_CRn[CIE]=1) and enabling the specific type of interrupt in the count and control register: end of transfer (DMA2P40_CTn[ETIE]), terminal count (DMA2P40_CTn[TCIE]), and error (DMA2P40_CTn[EIE]). When an interrupt occurs for a given channel, the DMA controller sends a signal to the Universal Interrupt Controller. For the PPC440 CPU to take an exception, interrupts from the particular DMA channel must be enabled in the interrupt controller's interrupt enable register (UIC0_ER[UDMAn]=1). Also, the CPU's machine state register's interrupt enable bit must be enabled for the appropriate interrupt type (critical or non-critical), MSR[EE,CE]. See *Universal Interrupt Controller* on page 205 for more information on interrupt controller processing.

When the DMA controller generates an interrupt, the interrupt remains active until the appropriate bits are cleared in the DMA Status Register (DMA2P40_SR). In addition, interrupts from a channel performing a scatter/gather transfer cause the channel to pause until the interrupt is cleared.

24.1.8 Scatter/Gather Transfers

With a normal DMA transfer it is necessary to program a channel's control, count and control, source, and destination registers for each transfer. The scatter/gather capability of the DMA controller provides a more efficient solution for applications that require multiple transactions on a single DMA channel. Instead of individually programming a channel's registers, software creates a set (linked list) of descriptor tables in system memory. *Figure 24-12* illustrates the required table format.

Figure 24-12. Scatter/Gather Descriptor Table



The most significant bit of word 2, the transfer count and configuration register value, is the link (LK) bit. Bit number 1 (SGL) of the same word is the scatter/gather locator bit. It determines the location of the next scatter/gather table (0=PLB space, 1=OPB space). Other than these bits, the contents of the descriptor table are identical to the associated register definitions.

To configure a channel for a scatter/gather transfer the DMA Scatter/Gather Descriptor Address Registers (DMA2P40_SGHn and DMA2P40_SGLn) for the channel are set to the address of the first descriptor table, which must be quad word (16 byte) aligned. The linked list of descriptor tables for a given programming of the DMA controller must not cross between PLB or OPB address space. The location of the linked list is programmed into DMA2P40_SGC[SGLn].

To begin a scatter/gather transfer, software writes DMA2P40_SGC with the enable mask (DMA2P40_SGC[EMn]) set, the start scatter/gather bit (DMA2P40_SGC[SSGn]) set and indicates the location (PLB or OPB) of the descriptor tables via DMA2P40_SGC[SGLn]. The DMA controller then reads the descriptor table at address (DMA2P40_SGHn || DMA2P40_SGLn) and updates the DMA controller registers as shown in *Table 24-5*. Upon receiving the data from the scatter/gather descriptor table, the channel's terminal count status bit (DMA2P40_SR[TCn]) and end of transfer status bit (DMA2P40_SR[EOTn]) are automatically cleared.

Table 24-5. DMA to PLB 4 Registers Loaded from Scatter/Gather Descriptor

Descriptor Table Entry	Register Loaded
Channel Control Word	DMA2P40_CRn
Source Address	DMA2P40_SAHx and DMA2P40_SALn

Table 24-5. DMA to PLB 4 Registers Loaded from Scatter/Gather Descriptor (continued)

Descriptor Table Entry	Register Loaded
Destination Address	DMA2P40_DAHn and DMA2P40_DALn
Count and Control	DMA2P40_CTn
Next Descriptor Address	DMA2P40_SGHn and DMA2P40_SGLn

After loading the channel's registers from the descriptor table, the transfer functions as a normal non-scatter/gather operation.

If the LK (link) bit was not set the scatter/gather process stops when the current transfer completes. Otherwise, the DMA controller reads the descriptor table at address (DMA2P40_SGHn || DMA2P40_SGLn) and the process repeats.

24.1.9 Programming the DMA2P40 Controller

Before the DMA controller can transfer data it must be configured, both globally and on a per-channel basis. Global settings include the DMA Polarity Register (DMA2P40_POL) and DMA Sleep Mode Register (DMA2P40_SLP). For most applications, these registers should be configured when the DMA controller is first initialized. To prevent spurious activity resulting from changing the active level for DMAReqn, DMAAckn, or EOTn[TCn], a channel's configuration in the Polarity Register should not be altered when the channel is enabled (DMA2P40_CRn[CE]=1).

Each channel has Control (DMA2P40_CRn), Count and Control (DMA2P40_CTn), Source Address (DMA2P40_SAHn and DMA2P40_SALn), Destination Address (DMA2P40_DAHn and DMA2P40_DALn), and Scatter/Gather Descriptor Address (DMA2P40_SGHn and DMA2P40_SGLn) registers. The type of DMA transfer determines which of these registers must be programmed and what causes the channel to start. In all cases, the terminal count (CSn), end of transfer (TSn) and error status (RIn) bits in the DMA Status Register (DMA2P40_SR) must be cleared or the channel will not start.

The programming information that follows assumes that the DMA controller is operating in non-scatter/gather mode. To use scatter/gather transfers the channel configuration data must be written into a set of descriptor tables in system memory. See *Scatter/Gather Transfers* on page 622 for additional details.

24.1.9.1 Memory-to-Memory Transfers

Memory-to-memory transfers can be initiated either by software or by an external device. If initiated via software, the transfer begins as soon as the channel is configured and enabled. When initiated by hardware (also known as a device-paced memory-to-memory transfer), software configures the channel for a memory-to-memory move and transfers begin when an external device places an active request on the channel request line, DMAReqn.

Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers

To perform a device-paced memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P40_CRn[PW]) to the width of the device-paced memory.
2. Set the source (DMA2P40_SAHn and DMA2P40_SALn) and destination (DMA2P40_DAHx and DMA2P40_DALn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P40_CRn[PW]), otherwise an alignment error will occur.
3. In the count and control register:

Caution: Program the desired count in the transfer count, TC, field.

Caution: Optionally enable bursting, BRSTEN=1, and set the burst size field, BRSTSIZ, as desired.

4. Clear the channel's status bits in the DMA status register (DMA2P40_SR).
5. In the channel control register (DMA2P40_CRn):
   a. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
   b. If the device-paced memory is at the source memory location set PL=1.
   c. Set the source address increment, SAI, and destination address increment, DAI, as desired.
   d. Set the transfer mode to device-paced memory-to-memory, TM=0b11.
   e. If the memory controller is on the OPB set the source location bit, SL=1. If the controller is on the PLB, clear SL.
   f. Enable the channel, CE=1.

Once the DMA channel is configured for this mode, the external device initiates a transfer by activating the DMAReqn input. The PPC440EPx/GRx chip then reads the source memory, buffers the data in the DMA controller and then outputs the data to the destination memory address. Transfers continue as long as the controlling device maintains an active signal on DMAReqn and the channel count register (DMA2P40_CTn) is non-zero. To pause a device paced memory-to-memory transfer, the controlling device must deassert DMAReqn one PerClk cycle before the last cycle in the device-paced memory access.

Software-Initiated Memory-to-Memory Transfers (Non-Device Paced)

To perform a software-initiated memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P40_CRn[PW]) as desired.
2. Set the source (DMA2P40_SAHn and DMA2P40_SALn) and destination (DMA2P40_DAHn and DMA2P40_DALn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P40_CRn[PW]), otherwise an alignment error will occur.
3. Program the transfer count field in the control and count register (DMA2P40_CTn[TC]) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA2P40_SR).
5. In the channel control register (DMA2P40_CRn):
   a. Optionally enable the DMA buffer, BEN=1.
   b. Set the source address increment, SAI, and destination address increment, DAI, as desired.
   c. Set the transfer mode to software-initiated memory-to-memory, TM=0b10.
   d. If the source memory controller is on the OPB set the source location bit, SL=1. If the controller is on the PLB, clear SL.
   e. If the destination memory controller is on the OPB set the PL=1. If the memory controller is on the PLB clear the PL bit.
   f. Enable the channel, CE=1.

Once the channel is enable the DMA controller transfers data from source to destination until the channel count reaches zero. Note that memory-to-memory transfers initiated by software do not use DMAReqn or DMAAckn.

24.2 DMA to PLB3 Controller (DMA2P30)

DMA to PLB3 is attached to the 64-bit processor local bus. *Table 24-6* lists the DMA2P30 channel assignments in PPC440EPx/GRx.

Table 24-6. DMA to PLB 3 Channel Assignments

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 0	Sub-Channel 0	UART 0 Receive	
	Sub-Channel 1	UART 2 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 1	Sub-Channel 0	UART 0 Transmit	
	Sub-Channel 1	UART 2 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 2	Sub-Channel 0	UART 1 Receive	
	Sub-Channel 1	UART 3 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

Table 24-6. DMA to PLB 3 Channel Assignments (continued)

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 3	Sub-Channel 0	UART 1 Transmit	
	Sub-Channel 1	UART 3 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

24.2.1 External Interface Signals

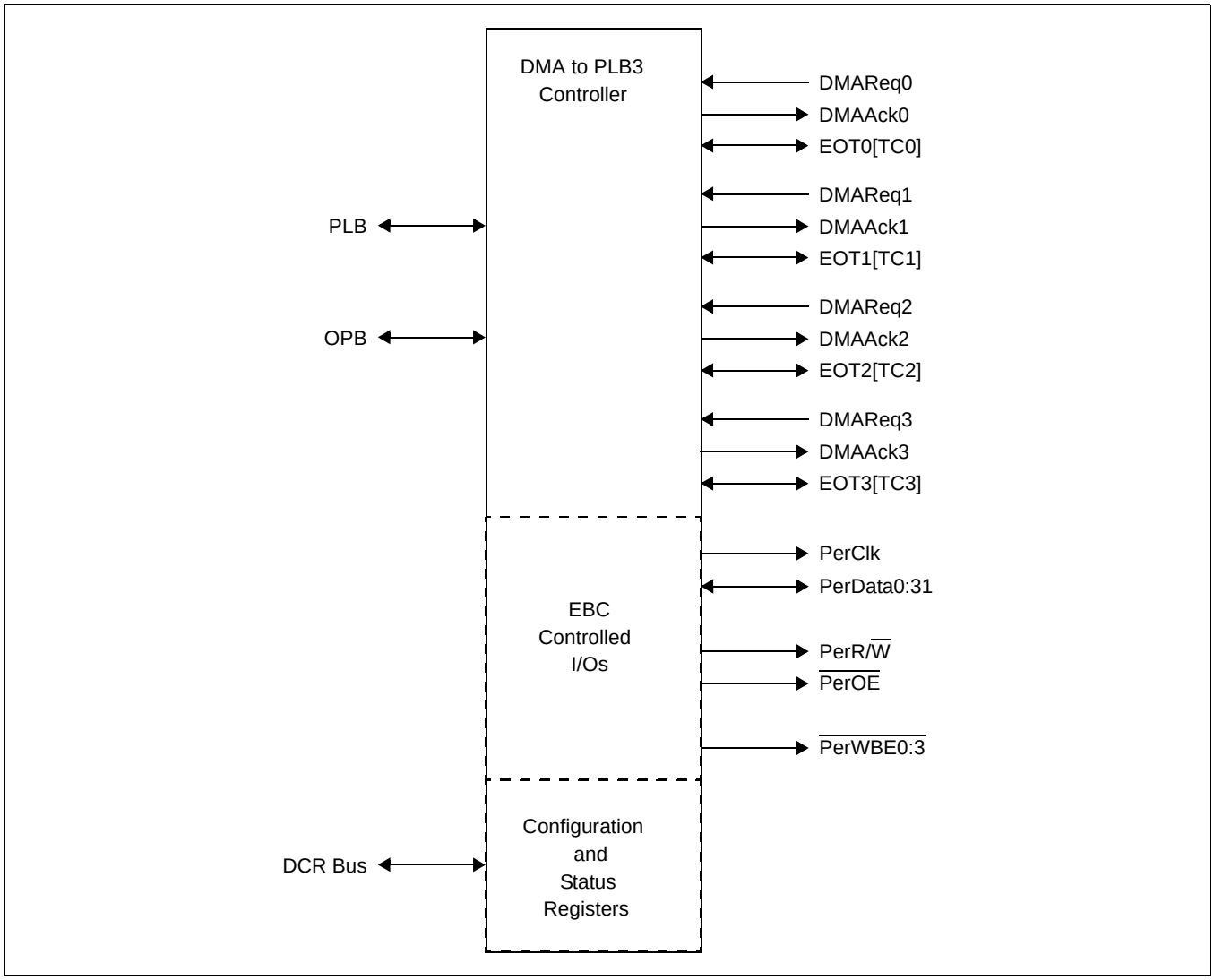
Figure 24-13 illustrates the external I/Os associated with the DMA controller and the EBC I/Os used during external peripheral transfers. External peripheral and EBC device-paced memory transfers request service from the DMA controller by driving a DMA request line (DMAReq0-3) active. For peripheral mode transfers the DMA controller acknowledges the request and transfers data by asserting a DMA acknowledge signal (DMAAck0-3). In contrast, an EBC device-paced memory-to-memory transfer occurs when the chip select (PerCSn) associated with the memory location is driven active. The timing of PerCSn and the other EBC I/Os is determined by the Bank Access Parameter Register (EBC0_BnAP) for the particular memory location. See *External Bus Controller* on page 555 for details on EBC configuration and timings.

Table 24-7. DMA to PLB3 Controller External I/Os

Signal	Usage
DMAReqn	DMA Request, used to request either a peripheral mode transfer or an EBC device-paced memory-to-memory transfer.
DMAAckn	DMA Acknowledge, instructs an EBC-attached DMA peripheral to transfer data.
EOTn[TCn]	End of Transfer or Terminal Count. Used to stop the channel when programmed as EOT. When configured as TC, goes active when the channel transfer count register (DMA2P30_CTn) reaches zero.

Note: The active level (polarity) of the DMAReqn, DMAAckn, and EOTn[TCn] signals is individually programmable. See *DMA to PLB 3 Polarity Configuration Register (DMA2P30_POL)* on page 630.. Operating information and timing waveforms are presented generically for active-high signals.

Figure 24-13. DMA to PLB 3 External Bus Controller Signals



24.2.2 Functional Overview

As a specialized controller, the DMA unit provides system designers and programmers with a highly efficient method of moving data. During any DMA transfer the controller always buffers data read from the source prior to writing the data to the destination. Since many buses, including the internal PLB and the SDRAM interface, provide substantially better performance when bursting data, the DMA controller includes a 32-byte (4 double words) buffer. This buffer is enabled on a per-channel basis by setting DMA2P30_CRn[BEN] and serves to minimize the number of discrete memory transactions. Each of the four DMA channels is configurable for either peripheral or memory-to-memory transfers.

24.2.3 Peripheral Mode Transfers

Peripherals are either devices attached to the EBC interface via the DMAReqn and DMAAckn lines, or the internal serial port (UART0). Memory is any address accessible from the PLB, including PLB-mapped PCI address space, SDRAM memory, and memory and memory-mapped devices on the peripheral bus. During a peripheral mode transfer the peripheral requests a DMA transfer by asserting a DMA request line. For UART0, this signal is internal

to the PPC440EPx/GRx, while external peripherals use one of the DMAReqn lines. When the requesting channel has the highest priority of any active channel, the requesting device receives a DMA acknowledge. In the case of external peripherals, the appropriate DMAAckn line is driven to the active state, with the timing specified in the DMA Control Register for the channel (DMA2P30_CRn).

There are two types of peripheral mode transfers: peripheral-to-memory and memory-to-peripheral. A peripheral-to-memory transfer reads data from a DMA device, while a memory-to-peripheral transfer writes data. In both cases, the peripheral interface never bursts and data is transferred at the width of the peripheral. If the DMA buffer is disabled for the active channel (DMA2P30_CRn[BEN]=0), each peripheral transfer causes a corresponding memory operation.

When buffering is enabled during a peripheral-to-memory transfer, data is collected until the 32-byte buffer is full, the peripheral deasserts DMAReqn, a higher priority DMA request becomes pending, or the channel completes. The buffer contents are then written to the target memory as efficiently as possible. If the initial programming of the channel's destination address register (DMA2P30_DAn) is 32-byte aligned, the buffer is emptied in one burst operation to the target memory.

Memory-to-peripheral transfers differ since the amount of data that will be requested by the peripheral is unknown. If the DMA buffer is disabled (DMA2P30_CRn[BEN]=0) a discrete source memory read occurs for each element in the DMA transfer. Since this is inefficient, the buffer should only be disabled for low data rate transfers or when the source memory is FIFO-like, and reads are therefore destructive. When the 32-byte buffer is enabled the controller uses the setting in DMA2P30_CRn[PF] to prefetch 1, 2, or 4 64-bit double words from the source memory. The DMA controller provides data from the buffer until the peripheral deasserts its request, the channel is interrupted by one of higher priority, or the transfer completes. Whenever any of these conditions occurs any unused data in the DMA buffer is discarded.

24.2.4 Memory-to-Memory Transfers

The DMA controller can perform either device-paced (hardware-initiated) or software-initiated memory-to-memory transfers. Device-paced memory transfers function identically to peripheral mode transfers, except that a chip select (PerCSn) serves as the DMA acknowledge instead of a DMAAckn output. As with peripheral mode transfers, bursts never occur on the device-paced side of the transaction.

Software-initiated memory-to-memory transfers between memories with fixed timings provide the best overall performance. During a software-initiated transfer the DMA controller knows the exact amount of data to be transferred. As a result, when the 32-byte DMA buffer is enabled (DMA2P30_CRn[BEN]=1) the controller uses bursts as much as possible. To ensure the highest bandwidth, source and destination addresses should be aligned on 32-byte boundaries.

There are three cases that limit the ability to burst during software-initiated memory-to-memory transfers. Bursting is not possible from the source memory when the source address increment is zero (DMA2P30_CRn[SAI]=0). Similarly, the DMA controller does not burst to the destination when the destination address increment is zero (DMA2P30_CRn[DAI]=0). Finally, the DMA controller cannot burst to or from any address that maps to a device-paced (EBC0_BnAP[RE]=1) EBC chip select (PerCSn).

24.2.5 Scatter/Gather Transfers

Each of the four DMA channels supports scatter/gather transfers. This scatter/gather capability allows the chaining of multiple DMA controller operations within a channel. During a normal DMA operation software must program the control, source address, destination address, and count registers for each transfer. Scatter/gather transfers differ in that these registers are automatically loaded from a linked list data structure in system memory. When a channel completes one transfer the DMA controller loads the next set of configuration values into the channel's registers and the channel continues with the new programming.

24.2.6 Configuration and Status Registers

Table 24-8 lists the DMA2P30 configuration and status registers, each of which is accessed using the **mtdcr** and **mfdcr** instructions. As example, the following assembly code writes DMA2P30_CR0 and then reads DMA2P30_SR:

```
#define DMA2P30_CR0  0x100
#define DMA2P30_SR   0x120

    mtdcr     DMA2P30_CR0,r3            ! write r3 to channel 0 control register
    mfdcr     r4,DMA2P30_SR             ! read contents of status register into r4
```

The DMA configuration and status registers are readable at any time. However, because each register read requires a separate operation, it is not possible to guarantee that the values read from multiple registers correspond to a state that ever existed in the DMA controller. To illustrate, consider software that reads the destination address for channel 0 (DMA2P30_DA0) and then the count for channel 0 (DMA2P30_CT0). If the DMA controller updates the count between these two operations, the values read differ from what is expected.

While reads can occur at any time, software must not write the configuration registers for any channel that is currently enabled (DMA2P30_CRn[CE]=1). The only exception is that a channel may be disabled by reading the channel control register, clearing the channel enable bit, and then writing the new value to the control register. Once a channel is disabled, all of its configuration registers may be reprogrammed as desired.

Table 24-8. DMA to PLB 3 Controller Configuration and Status Registers

Mnemonic	DCR Address	Access	Description	Page
DMA2P30_CR0	0x0100	R/W	DMA to PLB 3 Channel Control Register 0	632
DMA2P30_CT0	0x0101	R/W	DMA to PLB 3 Count Register 0	634
DMA2P30_DA0	0x0102	R/W	DMA to PLB 3 Destination Address Register 0	634
DMA2P30_SA0	0x0103	R/W	DMA to PLB 3 Source Address Register 0	634
DMA2P30_SG0	0x0104	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 0	635
DMA2P30_SC0	0x0107	Page	DMA to PLB3 Subchannel ID for Register 0	636
DMA2P30_CR1	0x0108	R/W	DMA to PLB 3 Channel Control Register 1	632
DMA2P30_CT1	0x0109	R/W	DMA to PLB 3 Count Register 1	634
DMA2P30_DA1	0x010A	R/W	DMA to PLB 3 Destination Address Register 1	634
DMA2P30_SA1	0x010B	R/W	DMA to PLB 3 Source Address Register 1	634
DMA2P30_SG1	0x010C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 1	635
DMA2P30_SC1	0x010F	Page	DMA to PLB3 Subchannel ID for Register 1	636
DMA2P30_CR2	0x0110	R/W	DMA to PLB 3 Channel Control Register 2	632
DMA2P30_CT2	0x0111	R/W	DMA to PLB 3 Count Register 2	634
DMA2P30_DA2	0x0112	R/W	DMA to PLB 3 Destination Address Register 2	634
DMA2P30_SA2	0x0113	R/W	DMA to PLB 3 Source Address Register 2	634
DMA2P30_SG2	0x0114	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 2	635
DMA2P30_SC2	0x0117	Page	DMA to PLB3 Subchannel ID for Register 2	636
DMA2P30_CR3	0x0118	R/W	DMA to PLB 3 Channel Control Register 3	632
DMA2P30_CT3	0x0119	R/W	DMA to PLB 3 Count Register 3	634

Table 24-8. DMA to PLB 3 Controller Configuration and Status Registers (continued)

Mnemonic	DCR Address	Access	Description	Page
DMA2P30_DA3	0x011A	R/W	DMA to PLB 3 Destination Address Register 3	634
DMA2P30_SA3	0x011B	R/W	DMA to PLB 3 Source Address Register 3	634
DMA2P30_SG3	0x011C	R/W	DMA to PLB 3 Scatter/Gather Descriptor Address Register 3	635
DMA2P30_SC3	0x011F	Page	DMA to PLB3 Subchannel ID for Register 3	636
DMA2P30_SR	0x0120	R/Clear	DMA to PLB 3 Status Register	631
DMA2P30_SGC	0x0123	R/W	DMA to PLB 3 Scatter/Gather Command Register	635
DMA2P30_ADR	0x0124	R/W	DMA Address Decode Register	636
DMA2P30_SLP	0x0125	R/W	DMA to PLB 3 Sleep Mode Register	631
DMA2P30_POL	0x0126	R/W	DMA to PLB 3 Polarity Configuration Register	630

24.2.6.1 DMA to PLB 3 Polarity Configuration Register (DMA2P30_POL)

DMA2P30_POL is used to set the polarity (active state) of the external DMA I/O signals: DMAReqn, DMAAckn, and EOTn[TCn]. As shown in *Figure 24-14*, if a bit in DMA2P30_POL is 0, the corresponding signal is active high, otherwise the signal is active low.

Whenever any of the EOT polarities are changed (DMA2P30_POL[EnP), software must subsequently clear the corresponding EOT status bits in DMA2P30_SR[TS0:3] before enabling the associated DMA channel. This is necessary to prevent a channel from being disabled because of an incorrect EOT status stored in the status bits.

Figure 24-14. DMA to PLB3 Polarity Configuration Register (DMA2P30_POL)

0	R0P	DMAReq0 Polarity 0 DMAReq0 is active high 1 DMAReq0 is active low	
1	A0P	DMAAck0 Polarity 0 DMAAck0 is active high 1 DMAAck0 is active low	
2	E0P	EOT0[TC0] Polarity 0 EOT0[TC0] is active high 1 EOT0[TC0] is active low	
3	R1P	DMAReq1 Polarity 0 DMAReq1 is active high 1 DMAReq1 is active low	
4	A1P	DMAAck1 Polarity 0 DMAAck1 is active high 1 DMAAck1 is active low	
5	E1P	EOT1[TC1] Polarity 0 EOT1[TC1] is active high 1 EOT1[TC1] is active low	
6	R2P	DMAReq2 Polarity 0 DMAReq2 is active high 1 DMAReq2 is active low	
7	A2P	DMAAck2 Polarity 0 DMAAck2 is active high 1 DMAAck2 is active low	
8	E2P	EOT2[TC2] Polarity 0 EOT2[TC2] is active high 1 EOT2[TC2] is active low	

9	R3P	DMAReq3 Polarity 0 DMAReq3 is active high 1 DMAReq3 is active low	
10	A3P	DMAAck3 Polarity 0 DMAAck3 is active high 1 DMAAck3 is active low	
11	E3P	EOT3[TC3] Polarity 0 EOT3[TC3] is active high 1 EOT3[TC3] is active low	
12:31		Reserved	

24.2.6.2 DMA to PLB 3 Sleep Mode Register (DMA2P30_SLP)

DMA2P30_SLP enables the DMA controller to enter sleep (low-power) mode and programs the number of PLB clock cycles to wait when the controller is idle before going to sleep. The DMA controller only goes to sleep when no DMA channels are enabled and no configuration or status (DCR) register operations are in progress. Reading or writing any of the DMA control or status register awakens the DMA controller.

To enable sleep mode, set DMA2P30_SLP[SME] and CPM0_ER[DMA]. When sleep mode is enabled and the DMA controller becomes idle, the 10-bit idle timer begins counting down from the programmed value. Only the upper 5 bits of the idle counter are programmable; the lower 5 bits are hard coded to 0b11111. Therefore, the minimum granularity of the idle timer is 32 PLB clock cycles. When the counter reaches 0, the controller is placed in sleep mode.

Figure 24-15. DMA to PLB3 Sleep Mode Register (DMA2P30_SLP)

0:4	IDU	Idle Timer Upper 0–31	Upper 5-bits of the idle timer.
5:9	IDL	Idle Timer Lower Hard coded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the clock and power management logic to put the DMA controller to sleep.
11:31		Reserved	

24.2.6.3 DMA to PLB 3 Status Register (DMA2P30_SR)

As shown in *Figure 24-16*, DMA2P30_SR provides status information for each of the DMA channels. Bits in DMA2P30_SR are set in hardware, and can be either read or cleared by software. Clearing is performed by writing a word to DMA2P30_SR containing a 1 in any bit position to be cleared and 0 in all other bit positions.

The terminal count status (DMA2P30_SR[CSn]), end of transfer status (DMA2P30_SR[TSn]), and error status (DMA2P30_SR[RIn]) must be cleared for a DMA channel to operate. If a scatter/gather operation generates an interrupt for any of these conditions, the channel pauses until software clears any associated status fields in DMA2P30_SR.

Figure 24-16. DMA to PLB3 Status Register (DMA2P30_SR)			
0:3	CS[0:3]	Channel 0–3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0.
4:7	TS[0:3]	Channel 0–3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA2P30_CRn[ETD]=0.
8:11	RI[0:3]	Channel 0–3 Error Status 0 No error 1 Error occurred	See "Errors" on page 637 for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	
28:31		Reserved	

24.2.6.4 DMA to PLB 3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3)

DMA2P30_CR0–DMA2P30_CR3 are used to configure and enable their respective DMA channels. Before a DMA channel can transfer data, the channel control, source address, destination address, and transfer count registers must be programmed. If a DMA channel is programmed for scatter/gather transfers (DMA_SGC[SSGn]=1) the DMA channel control register is automatically loaded from memory. For additional details, see *Scatter/Gather Transfers* on page 622.

Figure 24-17. DMA to PLB3 Channel Control Registers (DMA2P30_CR0–DMA2P30_CR3)			
0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts are generated for terminal count, end of transfer, and errors conditions. See *DMA Interrupts* on page 638.
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location 0 External peripheral (EBC) bus 1 OPB (UART0)	

4:5	PW	Peripheral Width/Memory alignment 00 Byte (8 bits) 01 Half word (16 bits) 10 Word (32 bits) 11 Double word (64 bits) memory-to-memory transfers only	Transfer width equals peripheral width for peripherals.
6	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address as shown in the next column	If set to 1 increment as follows: • 1, if the transfer width is a byte (8 bits) • 2, if the transfer width is a half word (16 bits) • 4, if the transfer width is a word (32 bits) • 8, if the transfer width is a double word (64 bit)
7	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address as shown in the next column	If set to 1 increment as follows: • 1, if the transfer width is a byte (8 bits) • 2, if the transfer width is a half word (16 bits) • 4, if the transfer width is a word (32 bits) • 8, if the transfer width is a double word (64 bit)
8	BEN	Buffer Enable 0 Disable DMA 32-byte buffer 1 Enable DMA 32-byte buffer	If BEN=0, discrete read and write operations occur for each data transfer.
9:10	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
11:12	PSC	Peripheral Setup Cycles 0–3	Number of PerClk cycles that the EBC peripheral bus is idle from the last peripheral bus transaction until DMAAckn is active. Used only for the peripheral side of peripheral mode transfers.
13:18	PWC	Peripheral Wait Cycles 0–63	DMAAckn remains active for PWC + 1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
19:21	PHC	Peripheral Hold Cycles 0–7	The number of PerClk cycles from the time DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only for the peripheral side of peripheral mode transfers.
22	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
23	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
24:25	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are prioritized by channel number; channel 0 has the highest priority. See *Channel Priorities* on page 636 for more information.
26:27	PF	Memory Read Prefetch Transfer 00 Prefetch 1 double word 01 Prefetch 2 double words 10 Prefetch 4 double words 11 Reserved	Used only during memory-to-peripheral and device-paced memory-to-memory transfers. To enable prefetching it is required that BEN=1.
28	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity checking for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers" on page 637.

29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30:31		Reserved	

24.2.6.5 DMA to PLB 3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3)

DMA2P30_SA0–DMA2P30_SA3 contain source addresses for memory-to-memory and memory-to-peripheral transfers. If a DMA channel is setup for scatter/gather transfers (DMA_SGC[SSGn]=1), the associated source address register is automatically loaded from memory. For additional details, see *Scatter/Gather Transfers* on page 638.

The source address must be aligned at the transfer width programmed in DMA2P30_CRn[TW]. Otherwise, the error bit (DMA2P30_SR[RIn]) is set for the channel and no transfer occurs. If the source address increment bit in the channel control register is set (DMA2P30_CRn[SAI]), the address is incremented by the transfer width after each data transfer. In contrast, if the channel is performing a memory-to-peripheral transfer and the address decrement bit is set (DMA2P30_CRn[DEC]=1), the address is decremented by the transfer width after each transfer.

Figure 24-18. DMA to PLB3 Source Address Registers (DMA2P30_SA0–DMA2P30_SA3)

0:31		Source address for memory-to-memory and memory-to-peripheral transfers.	

24.2.6.6 DMA to PLB 3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3)

DMA2P30_DA0–DMA2P30_DA3 contain the destination address for memory-to-memory and peripheral-to-memory transfers. When a DMA channel is configured for scatter/gather transfers (DMA_SGC[SSGn]=1), the destination address register is automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 638.

The destination address must be aligned at the transfer width programmed in DMA2P30_CRn[TW]. Otherwise, the error bit (DMA2P30_SR[RIn]) is set for the channel and no transfer occurs. If the destination address increment bit in the channel's control register is set (DMA2P30_CRn[DAI]) the address is incremented by the transfer width after each data transfer. However, if the channel is performing a peripheral-to-memory transfer and the address decrement bit is set (DMA2P30_CRn[DEC]=1), the destination address is decremented by the transfer width after each transfer.

Figure 24-19. DMA to PLB3 Destination Address Registers (DMA2P30_DA0–DMA2P30_DA3)

0:31		Destination address for memory-to-memory and peripheral-to-memory transfers.	

24.2.6.7 DMA to PLB 3 Count Registers (DMA2P30_CT0–DMA2P30_CT3)

DMA2P30_CT0–DMA2P30_CT3 contain the number of transfers left in the DMA transaction for their respective channels when EOTn[TCn] is programmed as a terminal count output. When EOTn[TCn] is programmed as an end-of-transfer input (DMA2P30_CRn[ETD]=0), DMA2P30_CTn continues to count down past zero until EOTn is asserted.

If a DMA channel is setup for scatter/gather transfers (DMA_SGC[SSGn]=1), the count register is automatically loaded from memory. For additional details see *Scatter/Gather Transfers* on page 638.

The value in the DMA count register is interpreted as the number of *transfers* of the width specified in DMA2P30_CRn[PW], *not* the total number of bytes. The maximum number of transfers is 64 K, and each transfer can be either 1, 2, 4, or 8 bytes as programmed in DMA2P30_CR[PW]. The maximum count of 64 K transfers is programmed by writing zero to DMA2P30_CTn.

Figure 24-20. DMA to PLB3 Count Registers (DMA2P30_CT0–DMA2P30_CT3)

0:15		Reserved	
16:31	NTR	Number of transfers remaining	

24.2.6.8 DMA to PLB 3 Scatter/Gather Descriptor Address Registers (DMA2P30_SG0–DMA2P30_SG3)

When a DMA channel is setup for scatter/gather transfers (DMA_SGC[SSGn]=1), the Scatter/Gather Descriptor Address Register (DMA2P30_SGn) contains the memory address of the next scatter/gather descriptor table. Prior to starting a scatter/gather transfer, software must write the address of the channel's descriptor table to DMA2P30_SGn. Once the scatter/gather transfer starts, DMA2P30_SGn is automatically updated from the descriptor table. For additional details see *Scatter/Gather Transfers* on page 638.

Figure 24-21. DMA to PLB3 Scatter/Gather Desc. Addr. Regs (DMA2P30_SG0–DMA2P30_SG3)

0:31		Address of next scatter/gather descriptor table.	

24.2.6.9 DMA to PLB 3 Scatter/Gather Command Register (DMA2P30_SGC)

DMA2P30_SGC[SSGn] are the start scatter/gather enable bits for channels 0 to 3. DMA2P30_SGC[EMn] are the corresponding enable mask bits for DMA2P30_SGC[SSGn]. Setting DMA2P30_SGC[EMn] = 1 causes the selected channel to begin a scatter/gather operation, while writing a 0 stops the scatter/gather operation. To start or stop a specific scatter/gather channel, the corresponding DMA2P30_SGC[EMn] bit must be set to 1; otherwise, DMA2P30_SGC holds the previous value. Note that halting a scatter/gather transfer does not stop the transfer currently in progress.

Upon completion of a scatter/gather sequence of transfers, the DMA controller clears DMA2P30_SGC[SSGn].

If an error occurs when the DMA controller is reading the scatter/gather descriptor table, DMA2P30_SGC[SSGn] is cleared for the affected channel, and the channel error status bit (DMA2P30_SR[RIn]) is set.

For additional details see *Scatter/Gather Transfers* on page 638.

Figure 24-22. DMA to PLB3 Scatter/Gather Command Register (DMA2P30_SGC)			
0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3. 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3. 0 Writes to SSG[n] are ignored 1 Allow writing to SSG[n]	To write SSG[n], EM[n] must be set. Otherwise, writing SSG[n] has no effect.
20:31		Reserved	

24.2.6.10 DMA to PLB 3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3)

The DMA Sub Channel ID is a 32-bit register that contains the 3-bit ID in bits 0–2. These bits are used to allow one DMA channel to support up to eight different peripheral devices. This register's contents are driven out of the core as primary outputs (O_DMA_subchnl_ID0:O_DMAsubchnl_ID3). External multiplexing uses these output signals to select the request and EOT signals from the active device. The value of this register can also be loaded during scatter/gather operations.

Figure 24-23. DMA to PLB3 Subchannel ID Registers (DMA2P30_SC0–DMA2P30_SC3)			
0:2	SCID	DMA Subchannel ID	
3:31		Reserved	

24.2.6.11 DMA to PLB 3 Address Decode Register (DMA2P30_ADR)

OPB addresses are decoded by the DMA by comparing the most significant nibble of the source or destination address to the contents of the DMA2P30_ADR register. The DMA2P30_ADR register is written by a move to DCR operation. Upon reset, its value is set according to the I_DMA_PGM_dmaadr pins, which are tied high or low by the system designer upon instantiation of the DMA in a system. The 8-bit address compare decode field is implemented in bits 0–7 of the 32-bit register.

Figure 24-24. DMA to PLB3 Address Decode Register (DMA2P30_ADR)			
0:7	ADRD	Address decode compare field	
8:31		Reserved	

24.2.7 Channel Priorities

The priority of DMA transfers is controlled on a per-channel basis by the channel priority field in the channel control register. *Table 24-9* shows the different priority settings for DMA2P30_CRn[PW].

Table 24-9. DMA to PLB 3 Transfer Priorities

DMA2P30_CRn[CP]	Priority Level
0b00	Low
0b01	Medium Low

Table 24-9. DMA to PLB 3 Transfer Priorities

DMA2P30_CRn[CP]	Priority Level
0b10	Medium High
0b11	High

These priorities serve two purposes. First, the DMA controller arbitrates among all actively requesting channels and selects the highest priority channel for service. If multiple channels request at the same priority, the arbiter selects the lowest numbered channel for service.

DMA2P30_CRn[CP] determines the priority of the internal PLB transactions that the DMA controller uses to read and write data.

24.2.8 Data Parity During DMA Peripheral Transfers

When DMA2P30_CRn[PCE] = 1, parity checking is enabled for peripheral mode transfers on channel n.

Note: The peripheral bus controller (EBC) does not support data parity checking.

24.2.9 Errors

The DMA controller detects and reports three types of errors: address alignment, PLB timeout and slave errors. The DMA controller reports errors through the channel error status bit in the DMA status register (DMA2P30_SR[RIn]). If the error status bit for a channel is set, the channel enable bit (DMA2P30_CRn[CE]) is cleared, disabling the channel. An interrupt signal is also presented to the interrupt controller if DMA2P30_CRn[CIE] is set. See *DMA Interrupts* on page 638 for more information on interrupt processing.

When the DMA controller has multiple channels active, an error may be reported on the current channel which was in actuality caused by a previously active channel. This causes the current channel to have its error status bit set. Therefore, for deterministic error analysis with multiple DMA channels active the PLB slave bus controller's error status registers (the bus error address register in particular), must be queried to isolate the actual channel which encountered the error. In any case, the channel causing the errors will eventually cause all active channels, including itself, to be disabled.

24.2.10 Address Alignment Error

The source address (DMA2P30_SAn) and destination address (DMA2P30_DAn) registers must be aligned to the programmed transfer width (DMA2P30_CRn[TW]).The address alignment rules are outlined in *Table 24-10*. In addition, when a channel is configured for scatter/gather transfers, the scatter/gather table must be word-aligned. If the source, destination and scatter/gather address registers are not appropriately aligned an error occurs immediately after the channel is enabled.

Table 24-10. Address Alignment Requirements

DMA2P30_CRn[PW] Setting	Required Alignment for DMA2P30_SAn and DMA2P30_DAn
0b00	Byte (8-bit)
0b01	Half word (16-bit)
0b10	Word (32-bit)
0b11	Double word (64-bit)

24.2.11 PLB Timeout

The DMA controller uses PLB operations to read and write memory. A PLB timeout results if the DMA controller attempts to access a non-existent memory location. This will occur if the source, destination or scatter/gather address registers do not map to valid memory locations.

24.2.12 Slave Errors

If the DMA controller detects an error from a PLB slave, it finishes any active read/write pair transfer on the channel and then reports an error. An SDRAM uncorrectable ECC error and an EBC bank protection error are examples of PLB slave errors.

24.2.13 DMA Interrupts

Each DMA channel can generate interrupts for end of transfer, terminal count and error conditions. Interrupts from a particular DMA channel are enabled by setting the channel enable bit in the channel's control register (DMA2P30_CRn[CIE]=1). When an interrupt occurs for a given channel, the DMA controller sends a signal to the Universal Interrupt Controller. For the PPC440 CPU to take an exception, interrupts from the particular DMA channel must be enabled in the interrupt controller's interrupt enable register (UIC0_ER[Dn]=1). Also, the CPU's machine state register's interrupt enable bit must be enabled for the appropriate interrupt type (critical or non-critical), MSR[EE,CE]. See *Universal Interrupt Controller* on page 205 for more information on interrupt controller processing.

For DMA channels with interrupts enabled (DMA2P30_CRn[CIE]=1) and not performing a scatter/gather transfer an interrupt is generated while any of the following are true:

DMA2P30_CRn[TCE]=1 and DMA2P30_SR[CSn]=1
DMA2P30_SR[TSn]=1
DMA2P30_SR[RIn]=1

When a channel is performing a scatter/gather transfer, interrupt generation is further qualified by the TCI, ETI and ERI bits loaded from the descriptor table (see *Table 24-11*). Any of the following conditions cause an interrupt during a scatter/gather transfer when interrupts are enabled for the channel (DMA2P30_CRn[CIE=1]):

TCI=1 and DMA2P30_CRn[TCE]=1 and DMA2P30_SR[CSn]=1
ETI=1 and DMA2P30_SR[TSn]=1
ERI=1 and DMA2P30_SR[RIn]=1

For both normal DMA and scatter/gather transfers the interrupt remains active until the appropriate bits are cleared in the DMA Status Register (DMA2P30_SR). In addition, interrupts from a channel performing a scatter/gather transfer cause the channel to pause until the interrupt is cleared.

24.2.14 Scatter/Gather Transfers

With a normal DMA transfer it is necessary to program a channel's control, source, destination, and count registers for each transfer. The scatter/gather capability of the DMA controller provides a more efficient solution for applications that require multiple transactions on a single DMA channel. Instead of individually programming a channel's registers, software creates a set (linked list) of descriptor tables in system memory. *Table 24-11* illustrates the required table format.

Table 24-11. Scatter/Gather Descriptor Table

Memory Address	Byte 0 (MSB)	Byte 1	Byte 2	Byte 3 (LSB)
x (word aligned)	DMA Channel Control Word			

Table 24-11. Scatter/Gather Descriptor Table

Memory Address	Byte 0 (MSB)						Byte 1	Byte 2	Byte 3 (LSB)
x + 4	Source Address								
x + 8	Destination Address								
x + 12	LK		TCI	ETI	ERI			Count	
x + 16	Next Scatter/Gather Descriptor Table Address								

Table 24-12 details the usage of the bit fields in the scatter/gather table.

Table 24-12. Bit Fields in the Scatter/Gather Descriptor Table

Bit	Mnemonic	Description
0	LK	Link 0 This is the last descriptor. 1 Fetch next descriptor from address DMA2P30_SGn when the channel completes
2	TCI	Enable Terminal Count Interrupt 0 Do not interrupt when terminal count occurs 1 Allow an interrupt when terminal count occurs
3	ETI	Enable End of Transfer Interrupt 0 Do not interrupt when end of transfer occurs 1 Allow an interrupt when end of transfer occurs.
4	ERI	Enable Error Interrupt 0 Do not interrupt if an error occurs 1 Allow an interrupt if an error occurs

To configure a channel for a scatter/gather transfer the DMA Scatter/Gather Descriptor Address Register (DMA2P30_SGn) for the channel is set to the address of the first descriptor table, which must be word-aligned. To begin the scatter/gather transfer, software then writes a start scatter/gather and enable mask to the Scatter/Gather Command Register (DMA2P30_SGC). The DMA controller then reads the descriptor table at address DMA2P30_SGn and updates the DMA controller registers as shown in *Table 24-13*. Upon receiving the data from the scatter/gather descriptor table, the channel's terminal count status bit (DMA2P30_SR[TCn]) is automatically cleared.

Table 24-13. DMA Registers Loaded from Scatter/Gather Descriptor Table

Descriptor Table Entry	Register Loaded
Channel Control Word	DMA2P30_CRn
Source Address	DMA2P30_SAn
Destination Address	DMA2P30_DAn
Count	DMA2P30_CTn
Next Descriptor Address	DMA2P30_SGn

After loading the channel's registers from the descriptor table, the transfer functions as a normal non-scatter/gather operation.

If the channel control word loaded from the descriptor table enables interrupts for the channel (DMA2P30_CRn[CIE]=1), the TCI, ETI, and ERI bits further qualify the generation of interrupts. See *DMA Interrupts* on page 638 for more information on scatter/gather interrupts.

If the LK (link) bit was not set the scatter/gather process stops when the current transfer completes. Otherwise, the DMA controller reads the descriptor table at address DMA2P30_SGn and the process repeats.

24.2.15 Programming the DMA to PLB3 Controller

Before the DMA controller can transfer data it must be configured, both globally and on a per-channel basis. Global registers include DMA2P30_POL and DMA2P30_SLP. For most applications, these registers should be configured when the DMA controller is initialized. To prevent spurious activity resulting from changing the active level for DMAReqn, DMAAckn, or EOTn[TCn], a channel configuration in tDMA2P30_POL should not be altered when the channel is enabled (DMA2P30_CRn[CE]=1).

The channel registers are DMA2P30_CRn, DMA2P30_SAn, DMA2P30_DAn, DMA2P30_CTn, and DMA2P30_SGn. The type of DMA transfer determines which of these registers must be programmed and what causes the channel to start. In all cases,DMA2P30_SR[CSn,TSn, RIn] must be cleared, or the channel will not start.

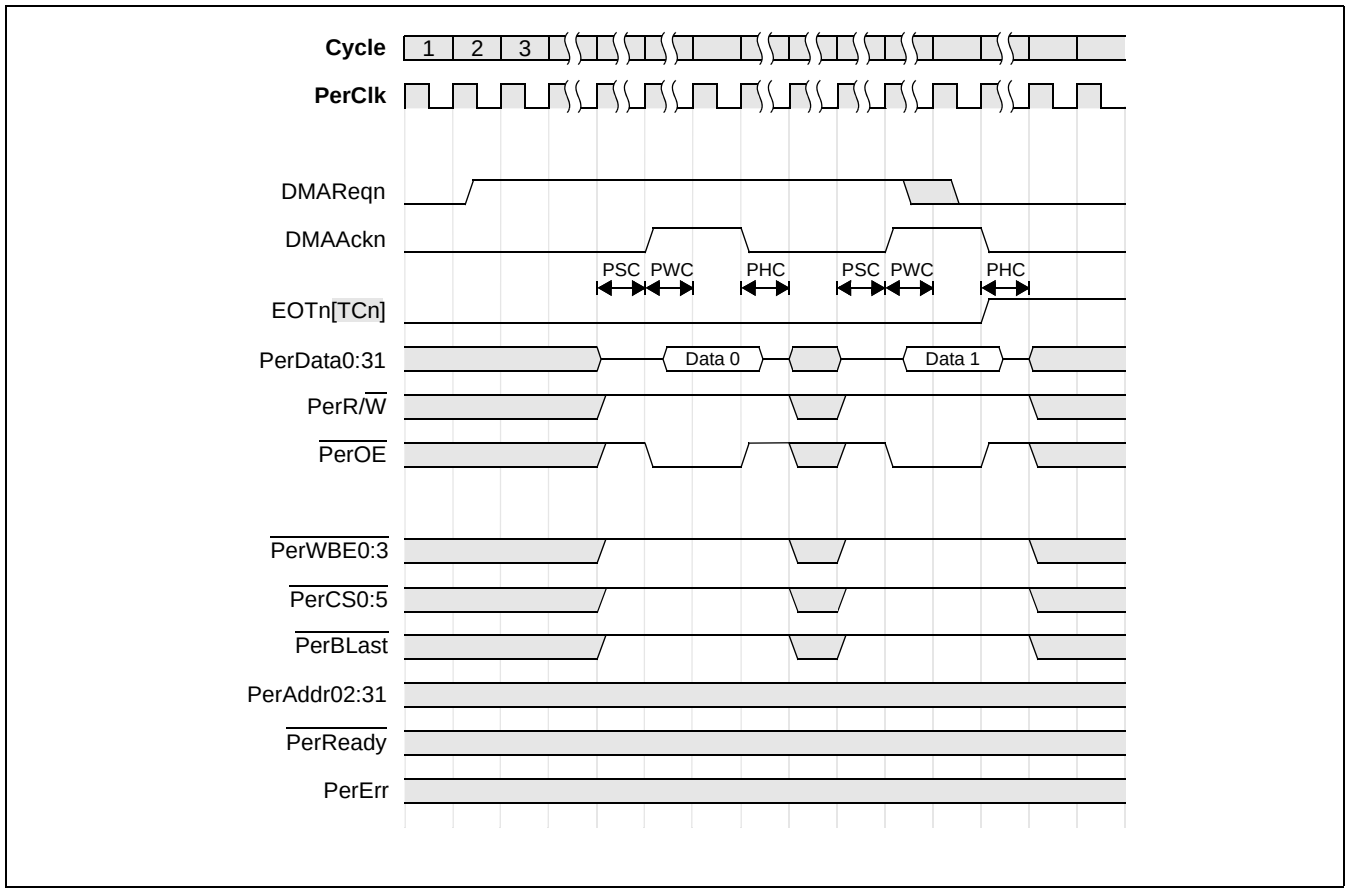
The programming information that follows assumes that the DMA controller is operating in non-scatter/gather mode. For scatter/gather transfers, the channel configuration data must be written into a set of descriptor tables in system memory, as described in *Scatter/Gather Transfers* on page 638.

24.2.16 Peripheral Mode Transfers

DMA peripherals are either devices attached to the EBC interface using the DMAReqn and DMAAckn lines, or the internal serial port (UART0). During a peripheral mode transfer an external peripheral asserts DMAReqn to request a DMA transfer. For metastability protection, DMAReqn is double-latched in the DMA on assertion, and sampled with a single latch on deassertion.

Timings on the memory-access portion of peripheral mode DMA transfers are governed by the configuration of the associated memory controller. In contrast, timing during the peripheral portion of the transfer is controlled by DMA2P30_CRn[PSC, PWC, PHC]. The effect of these parameters on peripheral timings is illustrated in *Figure 24-25* and *Figure 24-26*. Although shown as active high, the polarity (active state) of DMAReqn, DMAAckn, and EOTn[TCn] are programmable using DMA2P30_POL .

Figure 24-25. Peripheral to Memory DMA Transfers



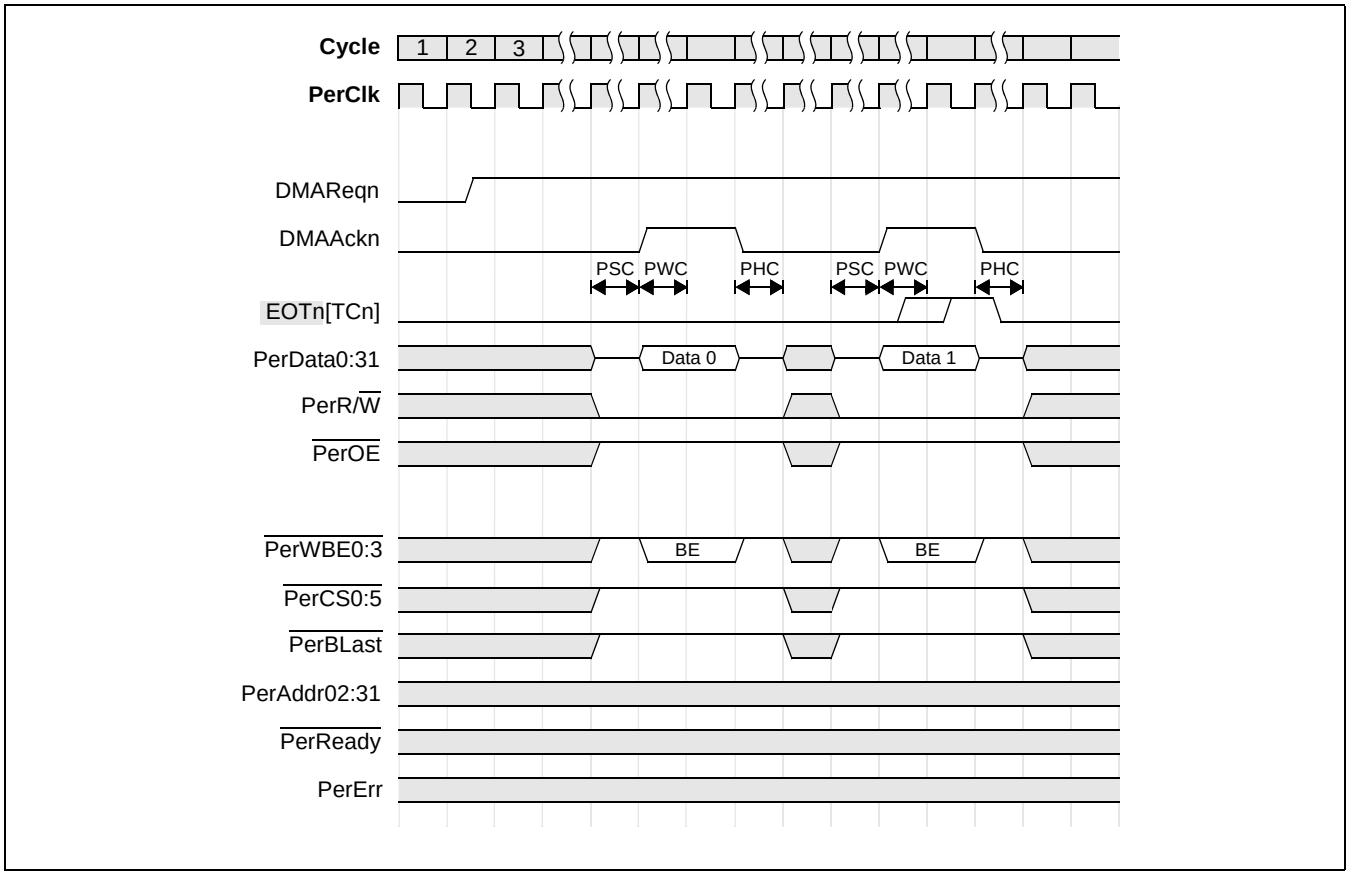
Peripheral setup cycles (DMA2P30_CRn[PSC]) provide a delay between any previous operation on the peripheral bus and DMAAckn becoming active. Following the setup time DMAAckn is driven active for (DMA2P30_CRn[PWC] + 1) PerClk cycles, until the rising edge of PerClk immediately before DMAAckn goes inactive. During peripheral-to-memory transfers read data from the peripheral is sampled on the rising PerClk edge where DMAAckn goes inactive. After DMAAckn becomes inactive the peripheral bus is held idle for DMA2P30_CRn[PHC] PerClk cycles.

The second transfer in *Figure 24-25* illustrates the required DMAReqn timing to prevent a subsequent DMA transfer. For all peripheral mode transfers DMAReqn must be sampled inactive at the end of the last PerClk cycle where DMAAckn is active.

The EOTn[TCn] I/O can be configured either as an end of transfer input (DMA2P30_CRn[ETD]=0) or a terminal count output (DMA2P30_CRn[ETD]=1). When programmed as a terminal count output, EOTn[TCn] is asserted in the cycle after DMAAckn became inactive and the channel's count register (DMA2P30_CTn) reached zero. EOTn[TCn] remains active until the terminal count status bit is cleared in the DMA status register (DMA2P30_SR[CSn]).

If EOTn[TCn] is configured as an end of transfer input (DMA2P30_CRn[ETD]=0), EOTn[TCn] must be sampled active during the last DMAAckn cycle. If the channel is configured for scatter/gather transfers EOTn[TCn] should be immediately deasserted to prevent the subsequent transfer from ending prematurely. *Figure 24-26* shows the required timing.

Figure 24-26. Memory to Peripheral DMA Transfers



For both peripheral-to-memory and memory-to-peripheral transfers the transfer width (DMA2P30_CR[PW]) must be set to the data bus width of the peripheral. This is because the DMA controller does not pack or unpack data on the peripheral side of transaction.

24.2.16.1 Peripheral-to-Memory Transfer

To perform a peripheral-to-memory DMA transfer from an EBC-attached DMA peripheral:

1. Set destination address register (DMA2P30_DAn) to the desired memory location. The address must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
2. Program the count register (DMA2P30_CTn) for the number of transfers.
3. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
4. In the channel control register (DMA2P30_CRn):
   a. Optionally enable the DMA buffer, BEN=1.
   b. Optionally enable parity checking, PCE=1.
   c. Set the destination address increment, DAI=1.
   d. Set the transfer mode to peripheral, TM=0b00.

e. Set the peripheral location to external, PL=0.
f. Set the transfer direction to peripheral-to-memory, TD=1.
g. Enable the channel CE=1.

Once the DMA channel is active, the peripheral initiates a transfer by activating the DMAReqn pin for the channel. The PPC440EPx/GRx then activates the DMAAckn pin to read data from the peripheral. This continues until either a terminal count or end of transfer condition occurs.

24.2.16.2 Memory-to-Peripheral Transfer

To perform a memory-to-peripheral DMA transfer to an EBC-attached DMA peripheral:

1. Set source address register (DMA2P30_SAn) to the desired memory location. The address must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
2. Program the count register (DMA2P30_CTn) for the number of transfers.
3. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
4. In the channel control register (DMA2P30_CRn):
   a. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
   b. Optionally enable parity generation, PCE=1.
   c. Set the source address increment, SAI=1.
   d. Set the transfer mode to peripheral, TM=0b00.
   e. Set the peripheral location to external, PL=0.
   f. Set the transfer direction to memory-to-peripheral TD=0.
   g. Enable the channel, CE=1.

Once the DMA channel is active, the peripheral initiates a transfer by activating the DMAReqn pin for the channel. The PPC440EPx/GRx then reads the source memory and subsequently activates the DMAAckn pin to write data to the peripheral. This continues until either a terminal count or end of transfer condition occurs.

24.2.17 Memory-to-Memory Transfers

Memory-to-memory transfers can be initiated either by software or by an external device. If initiated via software, the transfer begins as soon as the channel is configured and enabled. When initiated by hardware (also known as a device-paced memory-to-memory transfer), software configures the channel for a memory-to-memory move and transfers begin when an external device places an active request on the channel request line, DMAReqn.

24.2.17.1 Hardware-Initiated (Device-Paced) Memory-to-Memory Transfers

To perform a device-paced memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P30_CRn[PW]) to the width of the device-paced memory.
2. Set the source (DMA2P30_SAn) and destination (DMA2P30_DAn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
3. Program the count register (DMA2P30_CTn) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
5. In the channel control register (DMA2P30_CRn):

a. Optionally enable the DMA buffer, BEN=1, and set the desired prefetch count, PF.
b. If the device-paced memory is at the source memory location set PL=1.
c. Set the source address increment, SAI, and destination address increment, DAI, as desired.
d. Set the transfer mode to device-paced memory-to-memory, TM=0b11.
e. Enable the channel, CE=1.

Once the DMA channel is configured for this mode, the external device initiates a transfer by activating the DMAReqn input. The PPC440EPx/GRx then reads the source memory, buffers the data in the DMA controller and then outputs the data to the destination memory address. Transfers continue as long as the controlling device maintains an active signal on DMAReqn and the channel count register (DMA2P30_CTn) is non-zero. To pause a device paced memory-to-memory transfer, the controlling device must deassert DMAReqn one PerClk cycle before the last cycle in the device-paced memory access.

24.2.17.2 Software-Initiated Memory-to-Memory Transfers (Non-Device Paced)

To perform a software-initiated memory-to-memory DMA transfer:

1. Set the transfer width (DMA2P30_CRn[PW]) as desired.
2. Set the source (DMA2P30_SAn) and destination (DMA2P30_DAn) address registers to the desired memory locations. These addresses must be aligned to the programmed transfer width (DMA2P30_CRn[PW]), otherwise an alignment error will occur.
3. Program the count register (DMA2P30_CTn) for the number of transfers.
4. Clear the channel's status bits in the DMA status register (DMA2P30_SR).
5. In the channel control register (DMA2P30_CRn):
   a. Optionally enable the DMA buffer, BEN=1.
   b. Set the source address increment, SAI, and destination address increment, DAI, as desired.
   c. Set the transfer mode to software-initiated memory-to-memory, TM=0b10.
   d. Enable the channel, CE=1.

Once the channel is enable the DMA controller transfers data from source to destination until the channel count reaches zero. Note that memory-to-memory transfers initiated by software do not use DMAReqn or DMAAckn.

25. Memory Access Layer

The Memory Access Layer (MAL) is a hardware core that manages data transfers between packet-oriented communications cores, also known as EMACs (communications media access controllers) and memory. The PPC440EPx/GRx implements two Ethernet MACs which are managed by a single MAL. Only one MAL is required to handle all the data channels of the EMACs. Please note that EMAC is a generic term used to refer to the EMACs throughout the chapter.

To communicate with software device drivers, MAL utilizes a buffer descriptor ring structure in memory. A software device driver uses the buffer descriptor structure to inform MAL about buffer locations and packet or buffer status. MAL uses the buffer descriptors to convey packet transfer status from the EMAC core back to the software device driver. Each MAL channel requires its own buffer descriptor table ring structure in memory.

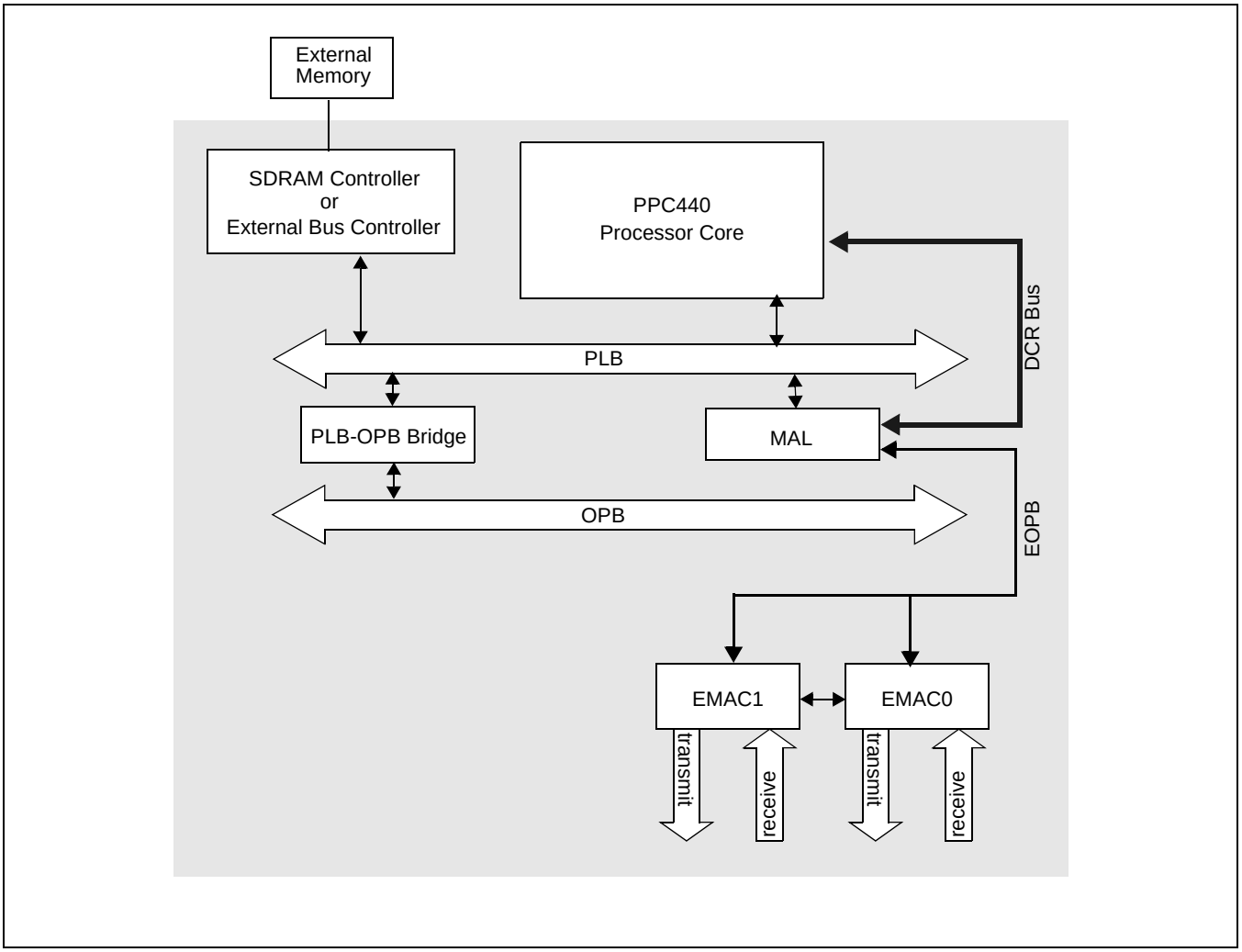
The PPC440EPx/GRx MAL manages the transfer of packets between the two Ethernet Media Access Controllers (EMAC0 and EMAC2) and the memory attached to the PPC440EPx/GRx (SDRAM or SRAM). The primary function of MAL is to move packets directly between memory and a EMAC core with minimal involvement of the processor.

25.1 MAL Features

- No restrictions on buffer alignment
- Aligned bus accesses to enable burst operation with external memories
- Configurable receive buffer size (configurable per channel)
- No minimum transmit buffer size
- Maximum buffer sizes of 4095 bytes (TX) and 4080 bytes (RX)
- Up to 256 descriptors in the buffer descriptor table per channel
- Configures EMAC according to commands specified in the descriptor status/control field
- Updates the descriptor status/control field at the end of packet transfer according to the status received from EMAC
- Buffer-based interrupt capabilities for each channel
- Concurrent operation of RX and TX channels
- Configuration using Device Control Registers (DCRs)
- Programmable PLB arbitration priority
- PLB/EOPB error detection

Figure 25-1 illustrates a general system structure overview of an embedded PowerPC processor core integrated with a packet oriented communication core.

Figure 25-1. General PPC440EPx/GRx Structure



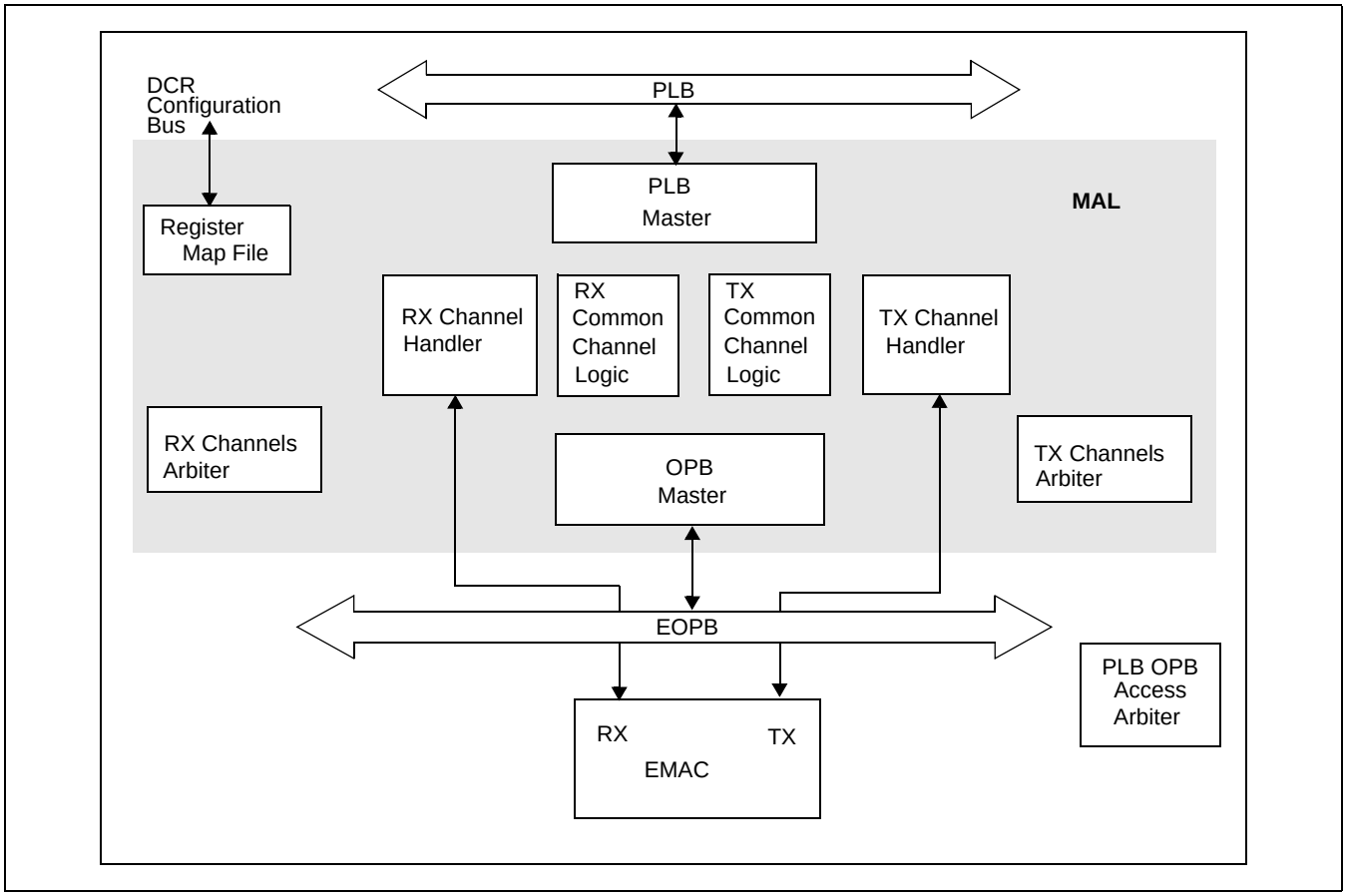
Two EMACs are configured and controlled by the processor core using the EOPB without MAL intervention. Packet data to be transmitted and received are stored in buffers in external memory. The MAL processes buffer descriptors and provides all data access facilities to the EMACs.

The MAL is not aware of EMACs such as EMAC as an entity. It is only aware of the EMAC's channels. In the PPC440EPx/GRx, each EMAC contains one TX channel and one RX channel. Transmit and receive operations can be performed simultaneously by MAL (full duplex). When a channel wins arbitration, MAL transfers data between system memory and the EMAC. MAL and the software driver maintain separate, dedicated buffer descriptor tables for each channel to maintain channel, packet, and buffer status. Packets can be constructed from one data buffer, or several data buffers (known as buffer chaining).

25.1.1 MAL Internal Structure

Figure 25-2 illustrates the MAL internal structure followed by a brief description of its contents.

Figure 25-2. MAL Internal Structure



25.1.1.1 PLB Master

The PLB master logic performs PLB transactions for the MAL, and is used to transfer data between a EMAC and memory, fetch buffer descriptors, and communicate status regarding data transfer.

25.1.1.2 EOPB Master

The EOPB master logic performs EOPB transactions for the MAL, and is used to transfer data between a EMAC and memory.

25.1.1.3 TX Channel Handler

The TX channel handler is a dedicated section for each TX channel. It keeps a record of the descriptor information and the current state of each channel.

25.1.1.4 RX Channel Handler

The RX channel handler is a dedicated section for each RX channel. It keeps a record of the descriptor information and the current state of each channel.

25.1.1.5 TX Channel Arbiter

The TX channel arbiter, connected to request lines from each TX channel, arbitrates between the TX channels and decides which channel gains access to the TX common channel logic.

25.1.1.6 RX Channel Arbiter

The RX channel arbiter, connected to request lines from each RX channel, arbitrates between the RX channels and decides which channel gains access to the RX common channel logic.

25.1.1.7 TX Common Channel Logic

The TX common channel logic is shared by all TX channels. It services a single TX channel at a time (selected by the TX arbiter). This logic activates the PLB and EOPB masters for data and buffer descriptor transactions.

25.1.1.8 RX Common Channel Logic

The RX common channel logic is shared by all RX channels. It services a single RX channel at a time (selected by the RX arbiter). This logic activates the PLB and EOPB masters for data and buffer descriptor transactions.

25.1.1.9 Register Map File

The register map file is used to configure MAL and read its status registers. Software accesses the MAL register file using the **mtdcr** and **mfdcr** instructions.

25.2 MAL0 Interfaces and Channel Assignments

MAL0 is comprised of four channels (two transmit and two receive). Each channel is dedicated to one of two EMAC cores. See *Table 25-1* for MAL0 channel assignments.

In the PPC440EPx/GRx, MAL0 uses 1/1 clocking, that is, the interface between MAL0 and the EMAC cores is clocked at the same frequency as the interface between MAL0 and the PLB.

Table 25-1. MAL0 Channel Assignment

MAL0 Channel	EMAC Channel
RX Channel 0	EMAC0 RX Channel
RX Channel 1	EMAC1 RX Channel
TX Channel 0	EMAC0 TX Channel
TX Channel 1	EMAC1 TX Channel

Sideband signals control slave selection and data transfer framing.

25.3 Transmit and Receive Operations

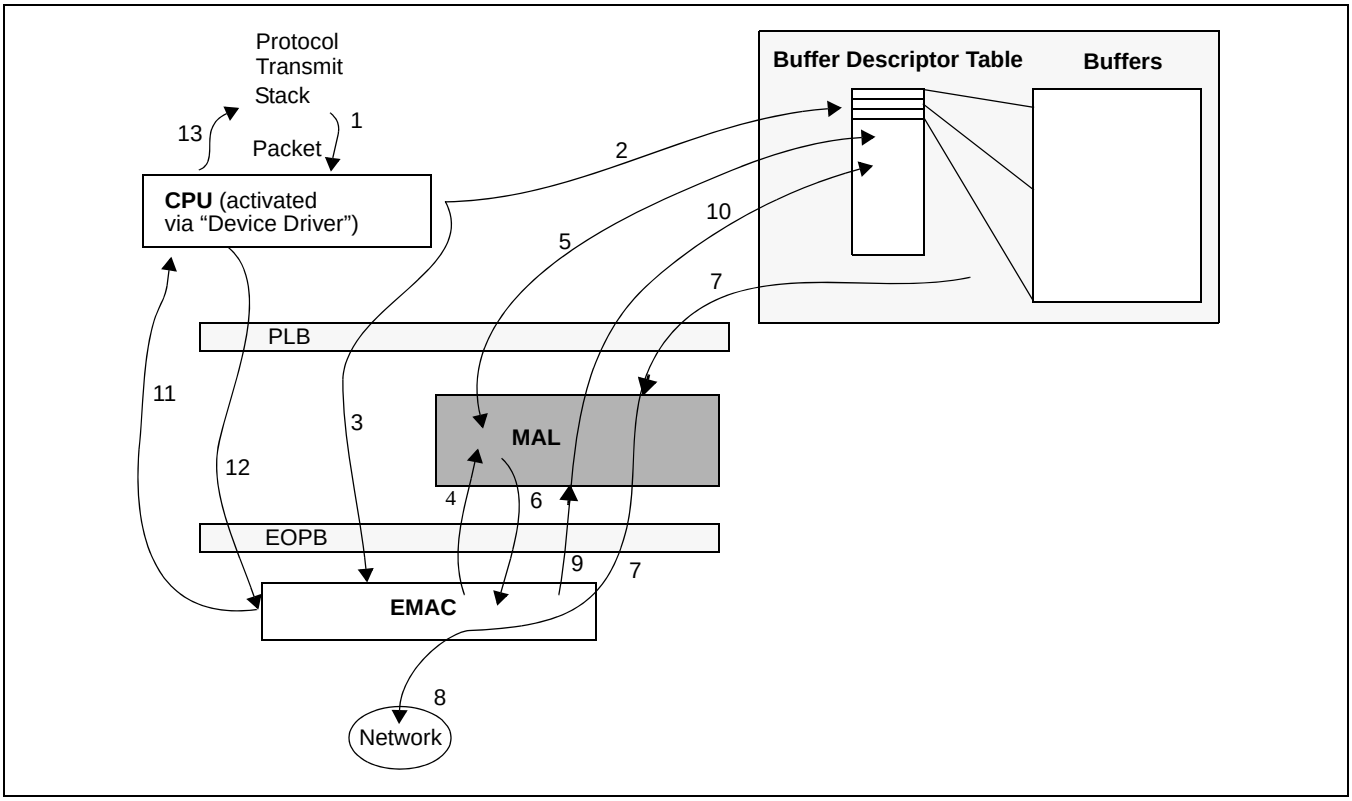
The device driver is responsible for configuring MAL before a EMAC can begin requesting MAL to process packets of data. The device driver should ensure that channels are not enabled during reconfiguration; otherwise, fatal errors may occur.

For more information about the MAL software interface, see *MAL Programming Notes* on page 662.

25.3.1 Transmit Operations

Figure 25-3 describes the software and hardware operations involved in a typical transmit operation.

Figure 25-3. Transmit Operations



The numbered steps above are described as follows:

1. The protocol stack (high-level software layer) initiates a packet transmit.
2. Software device driver parses the protocol stack buffer into descriptor table entries and buffers.
3. Software device driver instructs the EMAC to process a new transmit packet.
4. The EMAC channel requests MAL to process a new packet.
5. MAL fetches descriptor information.
6. MAL writes control information into the EMAC and initiates the data move.
7. Packet data is transferred from memory into the EMAC (the EMAC controls the pace of the data transfer).
8. The packet is transmitted on the media (steps 7 and 8 can overlap).
9. The EMAC requests that MAL read the packet status.
10. The status read by MAL is written back into a buffer descriptor.

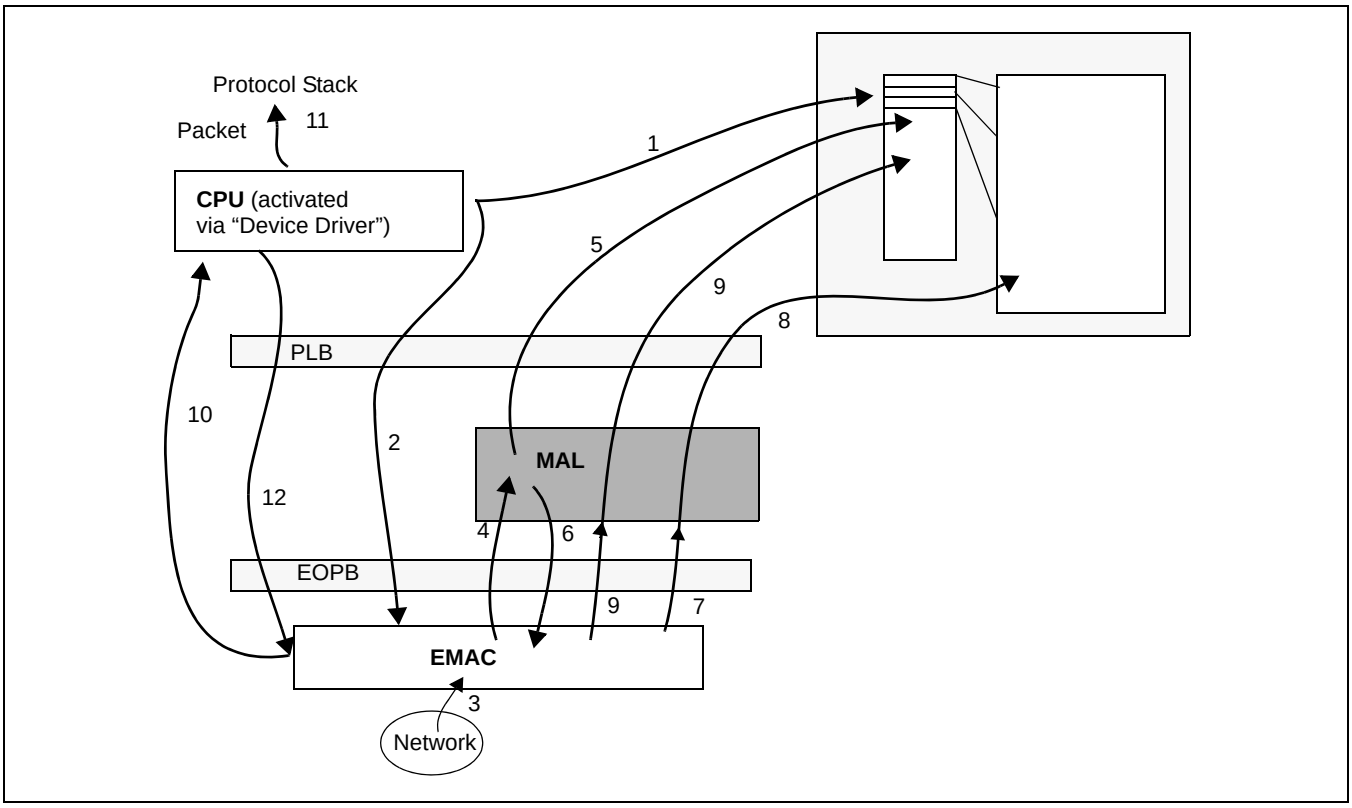
11. Software is interrupted (if interrupt conditions are met) by the EMAC or by the MAL end-of-buffer interrupt.
12. Software clears the interrupt status bits in the EMAC and in MAL.
13. Software informs the protocol stack that transmission is complete. (Typically, the stack assumes the packet will go once after step 1. In step 13 the software may return the buffer to the stack)

Note: The description in *Figure 25-4* is the general scheme for MAL operation in the software environment. The device driver should follow recommendations from *MAL Programming Notes* on page 662.

25.3.2 Receive Operations

Figure 25-4 describes the software and hardware operations when receiving a typical packet.

Figure 25-4. Receive Operations



The numbered steps above are described as follows:

1. Software device driver initializes the receive buffer descriptors.
2. Software device driver enables the EMAC to process a new packet.
3. A packet is received from the network (steps 2 and 3 can change in order).
4. The EMAC channel requests that MAL process a new packet.
5. MAL fetches receive buffer information from the descriptor table.
6. MAL writes the control word from the descriptor to the EMAC and initiates the data transfer.
7. The EMAC channel fills its FIFO storage.
8. MAL stores the packet in system memory buffers pointed to by the descriptors.
9. MAL reads status information from the EMAC and writes it to the buffer descriptors.

10. Software device driver is interrupted (if interrupt conditions are met) by the EMAC or by the MAL end-of-buffer interrupt.
11. The receive packet is passed to the protocol stack.
12. Software updates the receive buffer descriptor positions allowing them to be used again.

Note: The description in *Figure 25-4* is the general scheme for MAL operation in the software environment. The device driver should follow recommendations from *MAL Programming Notes* on page 662.

25.3.3 PPC440EPx/GRx System Device Control Registers

The following four registers are PPC440EPx/GRx system device control registers which are accessed indirectly through the SDR0_CFGADDR and SDR0_CFGDATA registers using the mtdcr and mfdcr instructions.

25.3.3.1 MAL Receive Burst Length Register (SDR0_MALRBL)

Figure 25-5. MAL Receive Burst Length Register (SDR0_MALRBL)

0		Reserved	
1:3	RBL0	0b000 Reserved 0b001 Reserved 0b010 8 words 0b011 16 words 0b100 32 words 0b101 64 words (default) 0b110 Reserved 0b111 Reserved	Indicates the requested amount of data to be received between MAL and EMAC0 in a single transaction.
4		Reserved	
5:7	RBL1	0b000 Reserved 0b001 Reserved 0b010 8 words 0b011 16 words 0b100 32 words 0b101 64 words (default) 0b110 Reserved 0b111 Reserved	Indicates the requested amount of data to be received between MAL and EMAC1 in a single transaction.
8:31		Reserved	

Note: For best performance, use the default values for all the burst length fields. Other valid values listed may be used at customer option.

25.3.3.2 MAL Receive Bus Size Register (SDR0_MALRBS)

Figure 25-6. MAL Receive Bus Size Register (SDR0_MALRBS)

0:3	RBS	MAL Receive Bus Size 0000 Reserved 1111 128-bit bus size	Indicates the OPB slave interface width for the EMAC attached to this channel
4:31		Reserved	

Note: Please use the default values for all the bits described in this register.

25.3.3.3 MAL Transmit Burst Length Register (SDR0_MALTBL)

Figure 25-7. MAL Transmit Burst Length Register (SDR0_MALTBL)

0		Reserved	
1:3	TBL0	0b000 Reserved 0b001 Reserved 0b010 8 words 0b011 16 words 0b100 32 words 0b101 64 words (default) 0b110 Reserved 0b111 Reserved	Indicates the requested amount of data to be transferred between MAL and EMAC0 in a single transaction.
4		Reserved	
5:7	TBL1	0b000 Reserved 0b001 Reserved 0b010 8 words 0b011 16 words 0b100 32 words 0b101 64 words (default) 0b110 Reserved 0b111 Reserved	Indicates the requested amount of data to be transferred between MAL and EMAC1 in a single transaction.
8:31		Reserved	

Note: For best performance, use the default values for all the burst length fields. Other valid values listed may be used at customer option.

25.3.3.4 MAL Transmit Bus Size Register (SDR0_MALTBS)

Figure 25-8. MAL Transmit Bus Size Register (SDR0_MALTBS)

0:3	TBS	MAL Transfer Bus Size 0000 Reserved 1111 128-bit bus size	Indicates the OPB slave interface width for the EMAC attached to this channel
4:31		Reserved	

Note: Use the default values for all of the bits described in this register.

25.4 Buffer Descriptor

The software interface for buffer descriptor (BD) processing consists of a set of registers within MAL and a set of circular queues in memory. Each transmit and receive EMAC channel has a descriptor table that contains buffer location and status information allocated to the channel. Each individual transmit or receive channel has its own buffer descriptor table. They are managed independently of each other. This section describes the individual transmit and receive interfaces.

Note: Since MAL uses a flat addressing scheme on the PLB, the physical memory can be located anywhere on the PLB address space. However, each data structure that the MAL works with (data buffers and buffer description tables) must not cross a 4GB address border.

During its operation, MAL is able to modify the contents of memory directly without processor core knowledge. Data cache coherency is the responsibility of the software device driver. To simplify device driver software, the MAL buffer descriptor tables should be placed in non-cached memory when possible. If this is not possible, the software driver must maintain cache coherency of the buffer descriptor tables by performing data cache flushes or invalidates when appropriate. When descriptors are in cached memory, the driver software must be aware that multiple descriptors are present in a single cache line and that cache invalidate or flush operations will be performed on multiple descriptors at the same time. This is significant because a cache line flush done by the driver to force a descriptor from the data cache to physical memory could corrupt another descriptor that occupies the same data cache line and is simultaneously being updated in physical memory by MAL.

Data buffers, in contrast, should be placed in cachable memory if possible. The software driver can easily maintain cache coherency of data buffers if:

- All buffers are aligned on a data cache line boundary
- All buffers are a multiple of a data cache line in size

Note: The data cache line size and alignment in the PPC440 CPU is 32 bytes.

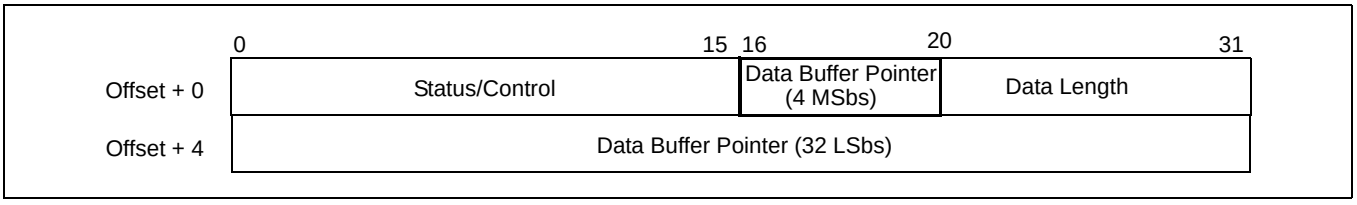
Before setting the empty bit in a RX buffer descriptor, the software must invalidate the memory occupied by the buffer in the data cache for the length specified in the RX buffer size register. Before transmitting a packet the software driver must flush the data buffer from the data cache before setting the Ready bit in the TX buffer descriptor.The software device driver or protocol stack fills the buffers pointed to by transmit buffer descriptors with packets to be transmitted, and/or provides empty buffers pointed to by receive buffer descriptors to be filled with received packets. Meanwhile, the hardware processes the descriptors, transfers the packet data to/from the EMAC, and updates the status fields of the descriptors.

Packet data associated with each transmit or receive channel is stored in buffers. Each buffer has an entry dedicated to it in one of the channel's buffer descriptor tables. MAL has a Channel Table Pointer Register for each of its channels. The EMAC (EMAC in the PPC440EPx/GRx) device driver sets the contents of these registers to point to the starting address of the buffer descriptor table for the associated channel.

The buffer descriptor table forms a circular queue with a programmable length. The last descriptor in the table is defined by setting the Wrap bit in the status/control field (see *Status/Control Field Format* on page 659). If there is no Wrap bit set in the table, then MAL automatically wraps after processing 256 descriptors (the maximum number of descriptors allow per channel).

The format of the buffer descriptor (BD) (see *Figure 25-9*) is the same for all EMACs, and has the same structure for both transmit and receive. The most significant halfword in each buffer descriptor contains a status/control halfword. This field contains two parts: the first part (6 bits) is BD handling information used by the MAL for descriptor processing, the second field (10 bits) is content specific for each EMAC. The second halfword determines the 4 MSbs of the data buffer pointer and the data length (in bytes) referenced in this buffer descriptor. The second word in the buffer descriptor contains the 32 LSbs of the data buffer pointer that points to the actual data buffer in memory. It is suggested that each data buffer start on a cache line boundary and be a multiple of a cache line in size if it resides in cachable memory. The cache line size in the PPC440EPx/GRx processor core is 32 bytes.

Figure 25-9. Buffer Descriptor Structure

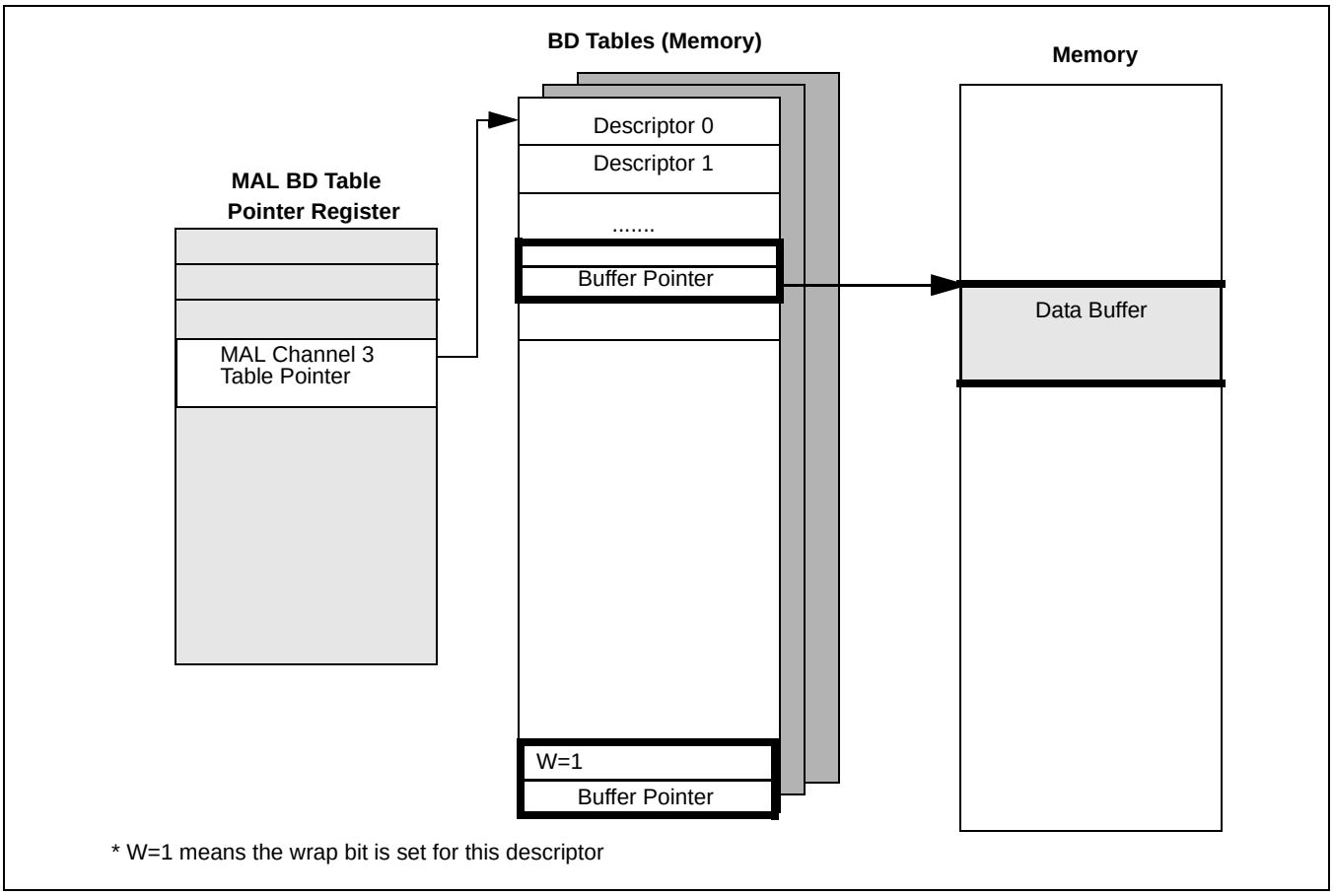


A packet may reside in as many buffers as necessary (transmit or receive). Each buffer has a maximum length of (4KB – 16) bytes. In TX channels, the buffer descriptor length field is written by the device driver and defines the number of bytes in the data buffer that is identified by the data buffer pointer. In RX channels, the buffer descriptor length field is written by MAL and defines the number of bytes written by MAL to the buffer that is identified by the data buffer pointer (see *Receive Software Interface* on page 657).

When processing a packet, MAL does not require that all buffers of the current packet are already valid. It requires the buffers to be ready in due time to be transmitted or received. Failure of the software to provide the descriptors in time may result in an error.

Figure 25-10 describes the structure of the packet in memory.

Figure 25-10. Packet Memory Structure



25.5 Transmit Software Interface

Once a channel is enabled in MAL (this is done by setting the appropriate bit in the Channel Active Register), a channel may request service from MAL. When the first transmit request comes in from a EMAC TX channel, MAL finds the starting address of the buffer descriptor table for the channel by looking in the corresponding Transmit Channel Table Pointer Register. If the first descriptor is marked as ready, MAL will start processing the associated buffer.

When MAL begins processing a packet, it writes the contents of the descriptor status/control field into the EMAC. This information, (depending on communication core implementation), may be used by the communication core to configure each packet transfer.

Once all data from the current buffer has been transferred to the communication core on the channel, MAL moves on to the next buffer descriptor in the table.

If a given buffer descriptor indicates that it contains the last section of the current packet, MAL informs the channel that the last data transferred to the channel completed the transfer of a data packet. At this point, the EMAC asks MAL to read the packet status. MAL then writes this information back into the status/control field of the last buffer descriptor of the packet.

The EMAC channel may request that MAL process the next buffer descriptor and the same packet handling process will be initiated. The first descriptor in the next packet follows the descriptor marked "last" in the previous packet.

25.5.1 Wrapping the BD Table for Transmit

When MAL processes a buffer descriptor (while handling a packet for a EMAC channel), it may encounter a Wrap indication within a buffer descriptor control field. This causes MAL to go back to the beginning of the buffer descriptor table for the next descriptor table entry. (This will also happen when MAL reaches the maximum number of descriptors.) The wrapping of the BD table, like all other BD table handling processes, is transparent to the EMAC.

25.5.2 Continuous Mode for Transmit

After transmitting the data pointed to by a buffer descriptor, MAL clears the Ready bit in the buffer descriptor control/status field. In this way, MAL will not process the same buffer descriptor again until software has filled the buffer with valid data and set the Ready bit in the descriptor again. While the Continuous Mode (CM) bit is set in the status/control field, MAL will not clear the Ready bit. The Continuous Mode allows re-transmission of the current data buffer without software intervention. This mode is generally used by protocols in which frequent re-transmission is an integral part of the protocol itself. In such cases, re-transmission can be performed without software intervention.

25.5.3 Back Up a Packet for Transmit

MAL is capable of re-transmitting the last packet ("back up a packet") following a request from a EMAC. If re-transmission is requested by the EMAC, it must be assured that all the buffers of the re-transmitted packet are available and were not re-processed by the device driver. In regular operation, MAL resets the Ready bit of each buffer descriptor when finished processing the descriptor. When MAL is requested by the EMAC to retransmit the last packet (the Back Up a Packet bit in the EMAC TX channel Status Halfword is set), MAL doesn't reset the READY bit in the last processed buffer descriptor, activate the end of packet interrupt, or write the status back to the descriptor in the memory. MAL also doesn't consider this as an end of packet event.

On the next service request from the same channel, MAL will start transmitting the packet again, starting from the first descriptor.

Note: The last processed buffer descriptor can be either the last descriptor of the packet or, in case of early packet termination, the buffer descriptor that was being processed when the transmit channel initiated the early packet termination. MAL will retransmit the backed-up packet regardless of the Ready bit value.

During retransmission of a backed-up packet, MAL may use descriptors on which the Ready bit was already cleared. Therefore, the device driver should not reuse descriptors before the Ready bit of the last descriptor is cleared.

Note: In the case of descriptor not valid, which is the first one in TX channel, EMAC is not allowed to return a status that contains a Back-Up a Packet request.

25.5.4 Descriptor Not Valid for Transmit

When MAL accesses a buffer descriptor, it checks whether or not the Ready bit is set. If the Ready bit is not set, two cases apply (special treatment of the READY bit is performed in the case of Back-up a packet):

For the case when the READY bit is not set:

- If the descriptor is the first descriptor of the packet MAL informs the channel that data is currently unavailable. Further handling of this scenario is EMAC-specific. The channel might either instruct MAL to access the same buffer descriptor periodically (by keeping its service request to MAL active) until it becomes ready, or 'give up' on the descriptor, completing the end-of-packet protocol with MAL. The channel might also indicate the buffer descriptor status to the device driver via an interrupt. However, in this case the EMAC should eventually complete the packet transfer protocol with MAL. Following a descriptor not valid indication, MAL's BD pointer continues pointing to the same location in the BD table. The next time a descriptor read is initiated by the EMAC, MAL will search for the buffer in the same location.
- If the descriptor is not the first descriptor of the packet, it is considered a descriptor error. MAL deactivates the channel and from its point of view, the processing of the current packet has ended. Software may learn about this situation from one of two MAL interrupts (or from both). The first one is a non-maskable interrupt that indicates the number of the TX channel, in which the descriptor error had occurred (interrupt bit for each TX channel, see *MAL Interrupt Enable Register (MAL0_IER)* on page 674). The second one is a maskable interrupt which indicates a descriptor error event, regardless the channel number (one interrupt bit for all the channels, see *MAL Error Status Register (MAL0_ESR)* on page 672). For more information about error handling, see *Error Handling* on page 663.

For the case of a back-up packet:

- When the current transmitted packet is a backed-up packet, all descriptors except the last, are valid even if the READY bit is not set. In this case, (not the last descriptor) MAL processes the packet descriptors regardless the READY bit value. If the READY bit of the last descriptor in the backed-up packet is not set, MAL treats it as a descriptor error. MAL handles the descriptor error as described above for the case when the packet isn't a backed-up packet.

25.5.5 Scroll Descriptors for Transmit

MAL may be configured by software, in the case of early packet termination, to scroll in the buffer descriptor table to the first descriptor of the next packet.

When a multiple-buffer packet is terminated early by the EMAC, while MAL is processing a buffer which is not the last buffer in the packet, MAL can operate in one of the following ways:

The MAL Scroll Descriptor in the configuration register is set:

- In this case MAL will read the status word from the EMAC channel. Then MAL will reset the READY bit in all the remaining buffer descriptors of the current packet. In addition, MAL will write the status to all the buffer descriptors. On the next service of this channel, MAL will fetch the first descriptor of the next packet.

The MAL Scroll Descriptor in the configuration register is clear:

- In this case MAL will read the Status word from the EMAC channel. Then MAL will terminate the current channel service by resetting the READY bit of the last processed buffer descriptor (the one in which there was an early termination) and will write the status only to this descriptor. On the next service of this channel, MAL will fetch the next descriptor in the current packet. In this case, the software is responsible to monitor the MAL location in the buffer descriptor table.

In the case that the EMAC requests a re-transmit of the early terminated packet (when the "backup" bit in the EMAC status is set), MAL will re-transmit the packet regardless of the MAL Scroll Descriptor bit.

25.6 Receive Software Interface

MAL uses the RX channel buffer descriptors in a manner similar to that used for transmission. Once an RX EMAC channel requests that a new packet be processed, MAL starts processing the channel's next buffer descriptor in the table. Once a channel is enabled in MAL, the channel may request MAL service. When it does, MAL accesses the first buffer descriptor (in that channel's buffer descriptor table) that is pointed to by the EMAC channel table pointer register. If that descriptor is ready (empty for RX), MAL will start processing the buffer.

When it begins processing each packet, MAL writes the contents of the status/control field into the EMAC. This information can be used by the EMAC for a per-packet configuration.

Once data is received from the memory, MAL moves the data from the RX channel FIFO into the data buffer pointed to by the first buffer descriptor. The current buffer descriptor may be closed for two reasons: there is no more room left in the buffer, or the EMAC channel indicated that the packet reception ended. If additional buffering space is needed for the current packet, MAL moves on to the next buffer descriptor. As each buffer descriptor is closed, MAL updates the length field with the actual amount of bytes written into the buffer. The maximal length of the buffers for each channel is defined by a configuration register (see *RX Channel Buffer Size Register (MAL0_RCBSx)* on page 677). The maximal receive buffer length is defined per channel.

Once the EMAC channel indicates that the packet reception has ended, it is expected to request that MAL update the received packet status in the BD status/control field. MAL updates the packet status and notifies the EMAC. At this point the packet is considered received and the EMAC may request that MAL begin the process of receiving a new packet. The first buffer of the next packet is the buffer in the BD table that followed the last descriptor of the previous packet.

25.6.1 Wrapping the BD Table for Receive

When MAL processes the buffer descriptor, it may encounter a Wrap indication within a buffer descriptor control field. This causes MAL to go back to the head of the channel's buffer descriptor for the next buffer descriptor. This also happens when MAL reaches the maximal number of descriptors.

25.6.2 Continuous Mode for Receive

After using a buffer descriptor, MAL sets the buffer descriptor control to the Not-Empty state. In this way, MAL will not use the same buffer descriptor a second time until the software has processed the not-empty buffer descriptor and set it to Empty again. MAL will not clear the Empty bit while the Continuous Mode (CM) bit is set in the status/control field. The Continuous Mode is generally used by protocols where frequent collisions are an integral part of the protocol itself (forcing the EMAC to abort a reception process and restart). In such cases, re-reception can be performed without software intervention.

25.6.3 Descriptor Not Valid for Receive

When MAL accesses a buffer descriptor it may find that the Empty bit is not set. In the case of an RX channel descriptor, this situation is considered as a descriptor error. MAL deactivates the channel and from its point of view, the processing of the current packet has ended. Software may learn about this situation from one of two MAL interrupts (or from both):

- An RXDE interrupt with the MAL0_RXDEIR indicating which channel caused the interrupt
- An SERR interrupt (system error) with one interrupt bit for all channels in the MAL0_ESR

For more about error handling, see *Error Handling* on page 663.

25.6.4 Buffer Length for Receive

The maximum length of an RX buffer descriptor is predetermined for all RX descriptors in each channel. The data-length value is programmable through a set of MAL registers (see *MAL Registers* on page 668). The actual data length field within the RX buffer descriptor is written by MAL. If the buffer is completely filled up, the value written will match the value programmed into the matching
RX-Channel-Descriptor data-length register. If the buffer is only partially filled up (for example, when the RX packet ended before running out of buffer space), the actual amount of space filled is written into the length field.

25.7 Buffer Descriptor Status/Control Fields

The following sections detail the status/control field bits. The information fields within the status/control field can be divided as follows:

- Information from a software device driver directed to MAL and EMAC
- Information from MAL and EMAC directed to software
- Status/control field handling
- Status/control field format
- TX status/control field format
- RX status/control field format

25.7.1 Information from a Software Device Driver Directed To MAL and EMAC

- MAL-related buffer descriptor processing information:
  - Buffer Ready/Not Ready (determines the buffer's validity).
  - Wrap to top of table or continue to next descriptor.
  - In a transmit buffer descriptor – Is the current buffer the last one in the packet?
  - Continuous or normal mode; that is, should MAL change the Ready/Not Ready value?
  - Should the channel generate an interrupt following the end of packet processing.
- EMAC channel configuration information:
  - Protocol specific configuration.

25.7.2 Information from MAL and EMAC Directed to Software

- MAL generated status information:
  - Buffer Ready/Not Ready (passes the buffer handling to software).
  - In receive buffer descriptor - Is the current buffer the first one in the packet?
  - In receive buffer descriptor - Is the current buffer the last one in the packet?
- EMAC channel generated status information:
  - Protocol specific error and status information (transmit and receive).

25.7.3 Status/Control Field Handling

When MAL accesses a new buffer descriptor, the status/control word is written to the EMAC channel. This allows the channel to configure itself for the current packet.

For all "intermediate" buffer descriptors (all descriptors that do not contain the packet's ending), the status/control field is written by MAL (rather than the EMAC). In this case, the status/control field indicates that the current buffer is not the last one in the current packet.

As MAL finishes processing the last buffer descriptor in a given packet, it reads the channel's status (via an EOPB transaction) and writes it into the buffer descriptor's status/control field.

In effect, since all of the various control and status fields do not overlap, the status/control halfword is read/written as a whole. Each agent (MAL, EMAC channel, and software) reads the entire status/control halfword, relates to specific fields of interest, and updates another subset of fields within the same halfword. While an agent modifies its related fields, all other fields remain unchanged.

25.7.4 Status/Control Field Format

The status/control halfword is divided into EMAC channel data and MAL related data. As explained above, the MAL related fields are either aimed at controlling MAL or written by MAL for use by the software. The MAL fields are of no interest to the EMAC (except the Ready and Empty bits).

The same applies to the EMAC channel fields. The EMAC related fields are either aimed at controlling the EMAC or written by EMAC for use by the software. These fields are of no interest to MAL.

MAL will not manipulate the EMAC related fields, and EMAC is not allowed to manipulate the MAL related fields.

25.7.5 TX Status/Control Field Format

The bit numbering in *Table 25-2* relates to the Buffer Descriptor's fullword which contains both the status/control and the length fields.

Table 25-2. TX Status Control Field

0	R	Ready 0 Not ready 1 Ready	This bit is set by the device driver and is cleared by MAL. The device driver sets this bit after preparing the buffer for transmission. MAL clears this bit when finish processing the buffer descriptor. MAL doesn't clear the Ready bit in the case of backing-up a packet request and in case of continuous mode (see *Back Up a Packet for Transmit* on page 655 and*Continuous Mode for Transmit* on page 655).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first descriptor buffer in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the EMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the R-bit is not cleared by MAL. This ensures that the data buffer is ready for transmission when MAL next accesses this buffer descriptor. However, the R-bit is cleared if an error occurs during transmission. This bit is controlled by software only. It controls MAL activities, and does not affect the EMAC channel.
3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is controlled by software only. It controls MAL activities, and does not affect the EMAC channel.
4		Reserved	It is assumed that this bit is set to zero by the software.
5	I	Interrupt 0 There is no action taken by MAL once it reaches the end of the current buffer. 1 After finishing processing the current buffer, if this bit is 1, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the EMAC.
6:15		EMAC Related Data	These bits are EMAC specific and may contain control fields generated by the software in order to control the EMAC channel. They may also contain status fields, generated by the EMAC channel, that will be processed by software.

25.7.6 RX Status/Control Field Format

The bit numbering in *Table 25-3* relates to the buffer descriptor's fullword which contains both the status/control and the length fields.

Table 25-3. RX Status Control Field

0	E	Empty 0 The data buffer associated with this buffer descriptor has been filled with received data, or data reception has been aborted due to an error condition. 1 The data buffer associated with this buffer descriptor is empty, or reception is currently in progress.	Software is free to examine or write to any fields of this buffer descriptor. While this bit is set to Not Empty, MAL will not use this buffer descriptor again. This buffer descriptor and its associated receive buffer are owned by MAL. Once the E-bit is set, software should not write to any fields of this RX buffer descriptor. MAL clears this bit after the buffer has been filled with received data or after an error is encountered. Software sets this bit to Empty after preparing the buffer for reception. This bit controls MAL and software activities (see bit 2: continuous mode below).
1	W	Wrap 0 This is not the last data buffer descriptor in the buffer descriptor table. 1 This is the last data buffer descriptor in the buffer descriptor table.	After this buffer has been used, MAL will transmit data from the first descriptor buffer in the table. This bit is controlled by software only. It controls MAL activities, and does not affect the EMAC channel.
2	CM	Continuous Mode 0 Normal Operation 1 Continuous Operation.	After this buffer descriptor is closed, the E-bit is not cleared by MAL. This ensures that the data buffer is ready to receive data when MAL next accesses this buffer descriptor. However, the E-bit is cleared if an error occurs during reception. This bit is controlled by software only. It controls MAL activities, and does not affect the EMAC channel.
3	L	Last 0 This is not the last buffer in the current packet. 1 This is the last buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
4	F	First 0 This is not the first buffer in the current packet. 1 This is the first buffer in the current packet.	This bit is updated by MAL following the activity of the channel.
5	I	Interrupt 0 There is no action taken by MAL once it reaches the end of the current buffer. 1 After finishing processing the current buffer, if this bit is 1, the end of buffer field in the End of Buffer Interrupt Status Register is set and the end of buffer interrupt is asserted.	MAL asserts the end of buffer interrupt after it updates the buffer descriptor's status field. This bit is controlled by software only. It controls the MAL activities and does not affect the EMAC.
6:15		EMAC Related Data	These bits are EMAC specific and may contain control fields generated by the software in order to control the EMAC channel. They may also contain status fields, generated by the EMAC channel, that will be processed by software.

25.8 MAL Programming Notes

The following sections contain information about programming the MAL.

25.8.1 MAL Initialization

MAL initialization includes two parts: configuration and channel activation.

Configuration involves two steps:

- MAL configuration - This step is done only after a power on reset or after a MAL soft reset. The following registers are involved:
  - Configuration Register (MAL0_CFG). This register defines MAL operation on the PLB and EOPB.
  - Interrupt Enable Register (MAL0_IER). This register is used to enable interrupts for various MAL error conditions.
- Channel specific configuration - This information may be changed only when the associated channel is not active. MAL0_TXBADDR and MAL0_RXBADDR must be written before writing to the MAL0_TXCTPxR amd MAL0_RXCTPxR registers. (The bit for the channel in the TX or RX Channel Active Set Register, is cleared.) The following registers are involved:
  - MAL0_RCBSx – RX Buffer Size (one register for each RX channel). This register defines the length of the RX buffers in memory.
  - MAL0_TXCTPxR or MAL0_RXCTPxR – Channel Table Pointer Register (one register for each channel). This register is programmed with the memory address of the first buffer descriptor table entry for the channel.

Setting the channel specific configuration can be done as part of MAL initialization or as part of the EMAC initialization process. In order to activate a channel, the following actions should be taken:

- The channel has to be configured in MAL
- The related bit in Channel Active Set Register (MAL0_TXCASR or MAL0_RXCASR) has to be set
- The channel operation must be enabled (EMAC configuration)

25.8.2 Interrupts

The MAL in the PPC440EPx/GRx has five interrupt lines which are connected to the Universal Interrupt Controller (*Universal Interrupt Controller* on page 205) Two interrupt lines, one for TX and one for RX, are used for interrupt events during packet transfer. An additional two interrupt lines, one for TX and one for RX, are used to report descriptor errors on a per-channel basis. The fifth interrupt is used to report MAL errors.

- TXEOB interrupt line is used to report end of buffer or end of packet for a specific TX channel. A bit for the related channel is set in the MAL0_TXEOBISR. See *End of Buffer Interrupt Status Registers* on page 671.
- RXEOB interrupt line is used to report end of buffer or end of packet for a specific RX channel. A bit for the related channel is set in the MAL0_RXEOBISR. See *End of Buffer Interrupt Status Registers* on page 671.
- TXDE interrupt line is used to indicate a descriptor error event in a specific TX channel descriptor table. A bit for the related channel is asserted in the MAL0_TXDEIR. See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 674.
- RXDE interrupt line is used to indicate a descriptor error event in a specific RX channel descriptor table. A bit for the related channel is asserted in the MAL0_RXDEIR. See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 674.
- SERR interrupt is used to report a system error indicated by MAL. For more information on handling the SERR interrupts, see *Error Handling* on page 663 and on page 672.

25.8.3 Error Handling

MAL handles errors on a per-channel basis. Within a EMAC channel, errors may arise from the EMAC (detected as an EOPB error), or from the memory access operations involved in MAL activity (detected as a PLB/descriptor error).

When a bus error occurs, MAL is notified by an EOPB or PLB error signal. EOPB errors are related to a specific channel and therefore channel operation is stopped. In the case of a PLB error, MAL cannot identify which channel is involved, therefore channel operation is not stopped. When a descriptor error occurs, MAL can again identify the channel involved, so channel operation is stopped. MAL stops channel operation by clearing the associated bit in the MAL0_TXCASR or MAL0_RSCASR register.

MAL keeps a record of the channels that experience errors and are made inactive. It also keeps a record of the characteristics of the first (or last) error detected (see *End of Buffer Interrupt Status Registers* on page 671).

Note: The device driver is responsible for configuring MAL before a EMAC can begin requesting MAL to process packets of data. The device driver should ensure that channels are not enabled during reconfiguration; otherwise, fatal errors may occur.

25.8.3.1 Error Detection

The MAL communication, both with EMACs and with memory, is carried out via the EOPB or PLB. As long as this bus communication is error-free and no descriptor errors are detected, MAL maintains normal activity with the channels set by the processor as active in the Channel Active Registers.

When an error is detected while performing a transfer for a channel, MAL asserts a maskable interrupt signal. If the identity of the channel is known (as is the case for EOPB errors or descriptor errors) then MAL immediately halts the dialogue with the channel. No further transactions are made, and that channel is registered by MAL as a non-active channel. MAL resets the channel by resetting its active bit in the Channel Active Register. Software must access the Channel Active Register in order to reactivate the channel.

If the identity of the channel that caused the error is not known (as is the case for PLB errors) then MAL continues to work normally. Error resolution and channel de-activation are the responsibility of the software.

25.8.3.2 Indicated Errors

Error description is stored in the Error Status Register (MAL0_ESR), (see *MAL Error Status Register (MAL0_ESR)* on page 672).

- Descriptor Error

  A descriptor error is a data error recognized during access to the descriptor table. The error can occur during TX or RX transmission.

  For RX channels, a descriptor error occurs when MAL accesses a descriptor in which the Empty bit is cleared.

  For TX channels, a descriptor error occurs when MAL accesses a descriptor in which the Ready bit is cleared. The following cases are exceptions.

  - On access to the first buffer descriptor in a TX packet.
  - On access to a buffer descriptor that is not the last descriptor in a backed-up packet.

  As a result of this error, the following actions are taken by MAL:

  - The Active bit of the related channel is reset and the channel activity is halted until software reactivates channel activity.

  - The associated bit in the TX Descriptor Interrupt Error Register (MAL0_TXDEIR) or RX Descriptor Error Register (MAL0_RXDEIR) is set, causing a non-maskable TXDE interrupt or RXDE interrupt respectively.
  - When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- EOPB Non-Fullword Error

  This error indicates that a non-fullword acknowledge was asserted by a slave.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until it is reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- EOPB Time-Out Error

  This error indicates that an EOPB time-out error was reported by the EOPB arbiter.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- EOPB Error

  This error indicates that an EOPB error was detected.

  Following this error, the active bit of the associated channel is reset and channel activity is halted until reactivated by software. When the channel is reactivated, MAL points to the descriptor at the head of the BD table.

- PLB Error

  This error indicates that a PLB error was detected (from the PLB slave).

  In this case, MAL cannot determine which channel caused the error. Therefore, operation is not halted for any of the channels.

25.8.3.3 Error Handling Registers

MAL error handling logic includes two registers.

- Error Status Register (ESR)

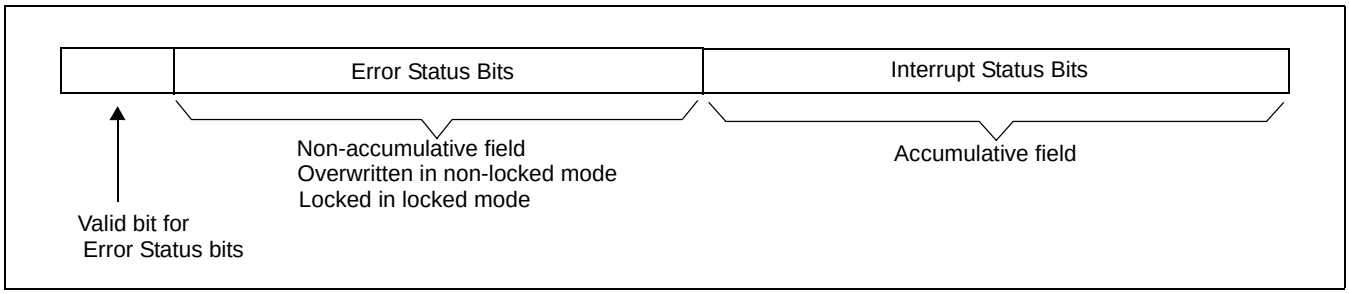
  This register holds information about the error that occurred and the interrupt status. The register includes the following fields:

  Error status – This field holds the error information. The information includes the number of the channel on which the error occurred (if known) and the type of the error. The error can be either the last detected error or a locked error if "Locked error mode" is active. See *Operational Error Modes* on page 665 for description of the Locked error mode.

  The error status field includes an "Error Valid" bit which indicates whether there is valid error information in the error status field or not. The error status field is not valid when the "Error Valid" bit is cleared (by writing 1 to this bit).

Interrupt status – Every error detected by MAL sets a related bit in the interrupt status field. Software can clear an interrupt status bit by writing 1 to the bit to be cleared. The bits in this field are accumulative which allows more than one interrupt to be indicated in the register.)

Figure 25-11. Error Status Register Field



25.8.3.4 Operational Error Modes

MAL can operate in two different error handling modes:

- Locked Error Mode: Information about the error is written to the Error Status Register, and the Valid bit in that register is set. Information in the Error Status field of the register stays locked until software unlocks it by resetting the error Valid bit. The Interrupt Status bits of the Error Status Register are not locked in this mode, so software can find out if more errors occur. However, the Error Status field applies only to the first error that is locked.
- Non-Locked Error Mode: Information about the error is written in the Error Status Register, and the error Valid bit is set. Each new error will be overwritten, so the information in the Error Status Field is valid only for the last error that occurred.

In both modes, each error written in the error description field will set the error Valid bit, and it is the responsibility of software to reset this bit.

The error handling mode is programmed in the MAL Configuration Register (see *MAL Configuration Register (MAL0_CFG)* on page 669).

25.8.3.5 Resolution of an Error Situation

When MAL encounters an error, it reacts as follows:

- Writes information about the error in the Error Status Register (ESR). This information includes the channel ID of the channel which caused the error (if known), the bus on which the error occurred, and the kind of error that occurred.
- Resets the channel that caused the error (if known) in the Channel Active Register.
- Updates the Interrupt Status bits in the MAL0_ESR. Then, depending on the mask defined in MAL0_IER (Interrupt Enable Register), it sends an interrupt to the Universal Interrupt Controller.

After receiving an interrupt from MAL, software can analyze the error information read from the Error Status Register. Software can restart channel activity by setting the associated bit in the Channel Active Register.

When a channel is stopped and restarted, MAL starts processing descriptors from the first descriptor in the channel descriptor table. Therefore, software may also update the value of the other channel related registers (see *Channel Table Pointer Registers* on page 675) in order to continue from the same buffer in memory.

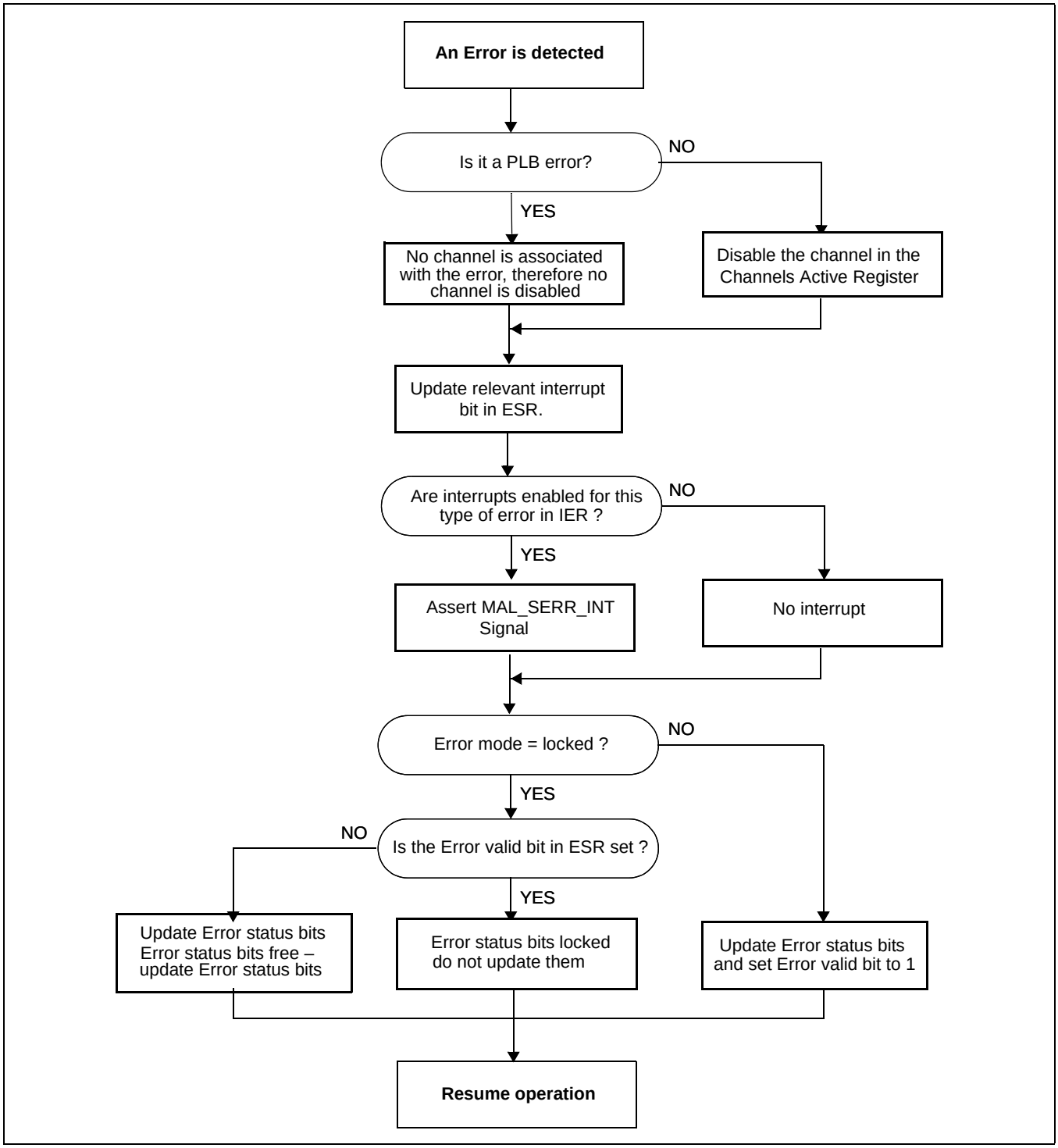
In the case of PLB errors, MAL does not know which channel caused the error. It is the responsibility of the software to analyze the MAL error registers and the PLB slave error registers to determine which channel caused the error. Software should reset the channel within MAL, resolve the problem, and then reactivate the channel.

See *Resolution of an Error Situation* on page 665 for a flow chart illustrating the steps MAL performs when resolving an error situation.

25.8.3.6 Interrupts To Software

Figure 25-12 describes MAL actions once an error is detected. Note that the actual decisions MAL makes may be in a different order than represented by this figure. In any case, the device driver should consider that all of the MAL actions are performed at the same time.

Figure 25-12. MAL Error Processing


An Error is detected
Is it a PLB error?
NO
YES
No channel is associated with the error, therefore no channel is disabled
Disable the channel in the Channels Active Register
Update relevant interrupt bit in ESR.
Are interrupts enabled for this type of error in IER ?
NO
YES
Assert MAL_SERR_INT Signal
No interrupt
Error mode = locked ?
NO
YES
NO
Is the Error valid bit in ESR set ?
YES
Update Error status bits Error status bits free – update Error status bits
Error status bits locked do not update them
Update Error status bits and set Error valid bit to 1
Resume operation

25.9 MAL Registers

The MAL registers are Device Control Registers (DCRs).

- Unless otherwise specified, all register fields are initialized at chip reset to 0.
- Reserved fields are read as undefined and must be written as 0s.

Table 25-4. MAL DCR Summary

Register	DCR Number	Access	Description	Page
MAL0_CFG	0x180	R/W	Configuration Register	669
MAL0_ESR	0x181	R/Clear	Error Status Register	672
MAL0_IER	0x182	R/W	Interrupt Enable Register	674
MAL0_TXCASR	0x184	R/W	Tx Channel Active Set Register	670
MAL0_TXCARR	0x185	R/W	Tx Channel Active Reset Register	670
MAL0_TXEOBISR	0x186	R/Clear	Tx End of Buffer Interrupt Status Register	671
MAL0_TXDEIR	0x187	R/Clear	Tx Descriptor Error Interrupt Register	674
MAL0_TXTATTRR	0x188	R/W	Tx PLB Attribute Register	675
MAL0_TXBADDR	0x189	R/W	Tx Descriptor Base Address Register	675
MAL0_RXCASR	0x190	R/W	Rx Channel Active Set Register	670
MAL0_RXCARR	0x191	R/W	Rx Channel Active Reset Register	670
MAL0_RXEOBISR	0x192	R/Clear	Rx End of Buffer Interrupt Status Register	671
MAL0_RXDEIR	0x193	R/Clear	Rx Descriptor Error Interrupt Register	674
MAL0_RXTATTRR	0x194	R/W	Rx PLB Attribute Register	675
MAL0_RXBADDR	0x195	R/W	Rx Descriptor Base Address Register	675
MAL0_TXCTP0R	0x1A0	R/W	Channel Tx 0 Channel Table Pointer Register	675
MAL0_TXCTP1R	0x1A1	R/W	Channel Tx 1 Channel Table Pointer Register	675
MAL0_RXCTP0R	0x1C0	R/W	Channel Rx 0 Channel Table Pointer Register	675
MAL0_RXCTP1R	0x1C1	R/W	Channel Rx 1 Channel Table Pointer Register	675
MAL0_RCBS0	0x1E0	R/W	Channel Rx Channel 0 Buffer Size Registers	677
MAL0_RCBS1	0x1E1	R/W	Channel Rx Channel 1 Buffer Size Registers	677

25.9.1 MAL Configuration Register (MAL0_CFG)

This register defines the operational mode of MAL. Unless a configuration change is required during system operation, the configuration register needs to be set only during system initialization.

Figure 25-13. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	
8:9	RPP	Read PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for read transactions.
10:11	RMBS	Read Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for read transactions. Determines the maximum data transfers (RdDAcks) per read burst transaction.
12:13	WPP	Write PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for write transactions.
14:15	WMBS	Write Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for write transactions. Determines the maximum data transfers (WrDAcks) per write burst transaction.
16	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.
17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the EMACs.
25:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). **Note:** An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.

30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode. Otherwise, it will handle errors in a non-locked mode.
31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related EMAC. When set, Scrolling mode is active.

25.9.2 Channel Active Set and Reset Registers

For the Channel Active Set/Reset Registers (MAL0_TXCASR, MAL0_TXCARR, MAL0_RXCASR, MAL0_RXCARR), each bit represents its associated channel (bit 0 for channel 0, and so on). When a bit is equal to 1, the channel has been enabled for operation. When a bit is equal to 0, the channel is disabled (MAL ignores any requests for service). If a channel is active when its enable bit is cleared, MAL stops processing the current packet. After the channel's enable bit is cleared, MAL goes back to the top of the channel descriptor table (pointed to by MAL0_TXCPTxR,MAL0_RXCPTxR).

- To enable a channel:
  - Write a 1 to its corresponding bit in the Channel Active Set Register (CASR).
  - Multiple channels can be enabled with a single CASR register write.
- To stop and reset a channel:
  - Write a 1 to its corresponding bit in the Channel Active Reset Register (CARR).
  - Writing a 0 to bits in the CARR registers has no effect on the channels.
  - Multiple channels can be reset with a single CARR register write.

MAL also clears the enable bit of a channel following an indication of an error on the channel. The CASR or CARR register(s) can be read to determine which channels are currently active.The following figures describe these registers.

Figure 25-14. TX Channel_Active Set Register (MAL0_TXCASR)

0:1	TCAS	Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two transmit channels.
2:31		Reserved	

Figure 25-15. TX Channel Active Reset Register (MAL0_TXCARR)

0:1	TCAR	Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has two transmit channels.
2:31		Reserved	

Figure 25-16. RX Channel_Active Set Register (MAL0_RXCASR)			
0:1	RCAS	Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has two receive channels.
2:31		Reserved	

Figure 25-17. RX Channel_Active Reset Register (MAL0_RXCARR)			
0:1	RCAR	Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has two receive channels.
2:31		Reserved	

25.9.3 End of Buffer Interrupt Status Registers

Each bit in the TX End-of-Buffer Interrupt Status and RX End-of-Buffer Interrupt Status registers is related to a channel's buffer descriptor table.

The TX End-of-Buffer Interrupt Status register contains the End-of-Buffer Status bits for each TX channel. The RX End-of-Buffer Interrupt Status register contains the End-of-Buffer Status bits for the RX channels. The mechanism (as described below) for both RX and TX registers is the same.

MAL sets a channel's bit in one of the following conditions:

- When MAL finishes the processing of a buffer (writes back the status to the current descriptor), the related bit in this register is set if the I bit in the descriptor status is set.
- When MAL finishes the processing of a packet (writes back the status of the packet's last buffer) and MAL0_MCR[EOPIE] is set.

  Note: In case MAL finishes the processing of a packet which is backed up, MAL doesn't consider it as an end of packet. Therefore, MAL will not set the appropriate channel bit in the End-of-Buffer Register.
- When the Bad Packet bit is set in the EMAC channel Status halfword.

The device driver resets the interrupt by writing a 1 to the related bit. Writing a 0 has no effect.

Figure 25-18. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)			
0:1	TCEI	Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two transmit channels.
2:31		Reserved	

Figure 25-19. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)			
0:1	RCEI	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

25.10 Error Registers

The following paragraphs describe MAL error registers. For more information about MAL errors see *Error Handling* on page 663.

25.10.1 MAL Error Status Register (MAL0_ESR)

This register holds the information about the error that occurred and the interrupts status. The register includes the following fields:

- **Error status bits** – This field holds the error information. The information includes the number of the channel on which the error occurred (if known) and the type of the error. The error can be either the last detected error or a locked error if "Locked error mode" is active. (See *Operational Error Modes* on page 665 for description of the Locked error mode.)

The error status field includes an "Error Valid" bit which indicates whether there is error information in the error status field or not. The error status bits are not valid when the "Error Valid" bit is cleared (by writing 1 to this bit).

- **Interrupt status bits** – Every error detected by MAL sets a related bit in the interrupt status field. The interrupt status bits may be cleared by software by writing 1 to the bit to be cleared. The bits in this field are accumulative (more than one interrupt may be indicated here). These bits are masked by the IER (Interrupt Enable Register) to create a maskable interrupt (which is the MAL_SERR interrupt in the UIC).

Note: In order to reset the interrupt bits and the Error valid bit in the Error Status register, 1 must be written to the related bit. Writing 0 has no effect.

More than one bit can be cleared at a time and only R/W bits can be reset.

Figure 25-20. MAL Error Status Register (MAL0_ESR)			
0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error. In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared. This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set
1	CIDT	Channel ID Type 0 CID field represents TX channel 1 CID field represents RX channel	Identifies current error as an Rx channel error or Tx channel error.

2:6	CID	Channel ID Number	Indicates the number of the channel that caused the error. **Note:** An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.
7		Reserved	
8	PTE	PLB Timeout Error 0 No error 1 PLB timeout error has occurred	Indicates the error is a PLB timeout
9	PRE	PLB Read Error 0 No error 1 PLB read error has occurred	
10	PWE	PLB Write Error 0 No error 1 PLB write error has occurred	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.
12		Reserved	
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout error has occurred	Indicates the error is an OPB timeout.
14	OSE	OPB Slave Error 0 No error 1 OPB slave error occurred	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error occurred	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:23		Reserved	
24	PTEI	PLB Timeout Error Interrupt 0 No error 1 PLB timeout error occurred	This bit is set following a PLB timeout error indication. Set condition for this bit generates a maskable interrupt.
25	PREI	PLB Read Error Interrupt 0 No error 1 PLB read error occurred	This bit is set following a PLB read error indication. Set condition for this bit generates a maskable interrupt.
26	PWEI	PLB Write Error Interrupt 0 No error 1 PLB write error occurred	This bit is set following a PLB write error indication. Set condition for this bit generates a maskable interrupt.
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is *not* the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28		Reserved	
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

25.10.2 MAL Interrupt Enable Register (MAL0_IER)

Each bit in the following register, when it is set, enables assertion of the MAL0 SERR interrupt in the UIC when the related bit in the MAL0_ESR (interrupt bit) is set.

Figure 25-21. MAL Interrupt Enable Register (MAL0_IER)

0:23		Reserved	
24	PT	PLB Timeout Interrupt 0 PLB timeout interrupt disabled 1 PLB timeout interrupt enabled	
25	PRE	PLB Read Error Interrupt 0 PLB read error interrupt disabled 1 PLB read error interrupt enabled	
26	PWE	PLB Write Error Interrupt 0 PLB write error interrupt disabled 1 PLB write error interrupt enabled	
27	DE	Descriptor Error Interrupt 0 Descriptor error interrupt disabled 1 Descriptior error interrupt enabled	
28		Reserved	
29	OTE	OPB Timeout Error Interrupt 0 OPB timeout error interrupt disabled 1 OPB timeout error interrupt enabled	
30	OE	OPB Error Interrupt 0 OPB slave error interrupt disabled 0 OPB slave error interrupt enabled	
31	PE	PLB Error Interrupt 0 PLB error interrupt disabled 1 PLB error interrupt enabled	

25.10.3 Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)

Each bit in the following registers is related to a channel buffer descriptor table. Each bit indicates a descriptor data error related to a certain channel.

The TX Descriptor Error register contains the descriptor errors bits of the TX channels. The RX Descriptor Error register contains the descriptor errors bits of the RX channels. The mechanism (as described below) for both RX and TX registers is the same.

MAL sets a channel's bit when a descriptor data error was recognized during access to the descriptor table of a specific channel (see *Descriptor Error* on page 663).

The device driver resets the interrupt by writing a 1 to the related bit. Writing a 0 has no effect. When one or more of the TX Descriptor Error Interrupt bits is set, then the MAL_TX_DESC_ERR_INT bit is set. When one or more of the TX Descriptor Error Interrupt bits is set, the MAL TXDE interrupt in the UIC is set. When one or more of the RX Descriptor Error Interrupt bits is set, the MAL RXDE interrupt in the UIC is set.

Figure 25-22. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)			
0:1	TXDE	Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL TXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has two transmit channels.
2:31		Reserved	

Figure 25-23. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)			
0:1	RXDE	Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL RXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has two receive channels.
2:31		Reserved	

25.10.4 PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR)

These registers define the specific transfer information driven by MAL towards the PLB slaves.

Figure 25-24. TX PLB Attribute Register (MAL0_TXTATTRR)			
0:15	PLBSA	TX PLB Storage Attributes	
16:31		Reserved	

Figure 25-25. RX PLB Attribute Register (MAL0_RXTATTRR)			
0:15	PLBSA	RX PLB Storage Attributes	
16:31		Reserved	

25.10.5 Channel Table Pointer Registers

Each channel has a full and separate 36-bit address space with the following limitations:

1. No data buffer is allowed to cross a 32-bit address space border.
2. No buffer descriptor table is allowed to cross a 32-bit address space border.

As the DCR bus is only 32 bits wide, each channel's table pointer needs to be configured in two DCR accesses as follows:

- Configure the 4 MSbs using a common register ((MAL0_TXBADDR or MAL0_RXBADDR))
- Configure the 32 LSbs using a channel private register (MAL0_TXCTPxR or MAL0_RXCTPxR)

25.10.6 Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)

Descriptor base address registers are used to update and read to/from all channels. MAL determines which channel should be used according to the specific TX/RX CTPR used. The flow of programming a table pointer address of 36 bits is described in the paragraphs that follow. Repeat the following steps for each channel.

Writing Flow

1. Write the four addresses' MSbs to the appropriate base address register (MAL0_TXBADDR or MAL0_RXBADDR).
2. Write the 32 LSbs to the specific channel's address register (MAL0_TXCTPxR or MAL0_RXCTPxR)
3. MAL will store the full 36-bit address combined from the above two DCR accesses in its internal memory.

Reading Flow

1. Read the 32 LSbs of the specific channel's address register.
2. MAL updates the (MAL0_TXBADDR or MAL0_RXBADDR) base address register according to the channel read in phase 1.
3. Read the 4 addresses' MSbs from bits 28 to 31 in the common base address register.

Figure 25-26. TX Descriptor Base Address Register (MAL0_TXBADDR)

0:27		Reserved	
28:31	DBA	Descriptor Base Address	

Figure 25-27. RX Descriptor Base Address Register (MAL0_RXBADDR)

0:27		Reserved	
28:31	DBA	Descriptor Base Address	

25.10.7 Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)

MAL uses RX Channel table pointer registers, one for each RX channel, and TX Channel table pointer registers, one for each TX channel. The Channel Table Pointer Registers point to the base address, in memory, of the buffer descriptor table used by each channel.

Note 1: When changing the value of any of the MAL0_TXCTPxR registers, the changed TX channels must be idle. To verify a channel is idle, check the device's Transmit Idle bit. Another way to assure that the channels are idle is to disable the channels before changing the MAL0_TXCTPxR register, and then re-enable them once the MAL0_TXCTPxR register is set to its new value.

Note 2: Although MAL supports a full 36-bit address space, no data buffer is allowed to cross a 32-bit address space border, and no buffer descriptor table is allowed to cross a 32-bit address space border.

The TX and RX Channel Table Pointer Registers have an identical format, as shown in *Figure 25-28*.and *Figure 25-29*. MAL0 has four TX channel table pointer registers, and four RX channel table pointer registers.

Figure 25-28. TX Channel Table Pointer Register (MAL0_TXCTPxR)			
0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.

Figure 25-29. RX Channel Table Pointer Register (MAL0_RXCTPxR)			
0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four receive channels.

The Table Pointer Registers retain their value following Soft Reset or Channel Reset.

25.10.8 RX Channel Buffer Size Register (MAL0_RCBSx)

Each RX channel has a fixed buffer length. The RX channel buffer size registers are configured by the device driver to determine the length of the buffer. This size is the maximal number of bytes that can be stored in the buffer. More than one buffer is used to store the incoming packet in case the length of the incoming packet data is more than the channel's buffer length (as defined by the channel's related register).

The buffer length for a given channel can be from 16 bytes to (4KB-16) bytes (in 16-byte steps). This length is represented by a 8-bit field. The buffer size in bytes is calculated as follows: RX_Buffer_Size_Field x 16.

Figure 25-30. RX Channel Buffer Size Register (MAL0_RCBSx)			
0:23		Reserved	
24:31	RCBS	Receive Channel Buffer Size	

25.11 Interrupt Coalescence

Interrupt coalescing allows users to tailor the behaviour of the Ethernet controller with respect to frame interrupts. Transmit and receive frames are handled separately with similar mechanisms. When coalescing is enabled (that is, when the bits 6, 7, 8, or 9 of UIC2 are set) transmit or receive frame interrupts are asserted when a predefined number of frames are transmitted or received, or a predefined time has elapsed, whichever occurs first.

There are two mutually exclusive ways to assert the interrupt:

1. The frame count threshold of transmitted or received frames has been exceeded, or
2. The timer has expired and at least one frame has been transmitted or received

In this implementation, the TX and RX functions are identical and present for the two channels of the MAL.

The interrupt generated is synchronous to the PLB clock and has a pulse width of one clock period.

25.11.1 Coalescing by Frame Counter Method

When consecutive packets arrive at the host processor, it is possible for the host to process interrupts only and not perform any other tasks. This can also lead to bandwidth congestion. By enabling coalescing, users can group frame interrupts, thus reducing the total number of interrupts to be processed. The frame threshold is defined in the interrupt coalescing configuration register. In a typical scenario, the threshold number can be anywhere in 1 to 255. Once the number of frames transmitted or received reaches the predefined threshold, an interrupt occurs, the counter is reset, and counting continues while the interrupt is active. The counter can also be reset when a pre-defined timer threshold is reached.

25.11.2 Coalescing by Timer Threshold

There can be a situation where no interrupt is raised for a long time and frames are waiting for processing. In this case, the user can define a timer threshold beyond which interrupts are forced. In the interrupt coalescing configuration register, the transmit and receive timer fields are set with reference to the number of interface clocks in the Ethernet block. This is different for 10Mbps, 100Mbps and 1Gbps Ethernet modes.

After transmitting a frame, the transmit interrupt coalescing threshold timer begins counting. When the threshold time is reached, an interrupt is raised if the appropriate bit is set. The receive operation is similar.

The timer can be accessed by software. It is readable at any time. It can also be cleared by software.

25.12 Interrupt Coalescence Registers

The following registers control interrupt coalescing.

25.12.1 Interrupt Coalescing Control Transmit Register (SDR0_ICCRTX)

SDR0_ICCRTX access is R/W.

Offset: 0x430B

Figure 25-31. Interrupt Coalescing Control Transmit Register (SDR0_ICCRTX)

0:8	FTHR0	Frame Count Threshold 0	Reset value of the field is 0b000000001. Frame count threshold for transmit side of MAL channel 0 .
9	FLUSH0	Flush with no Interrupt 0	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters without generating an interrupt, then write 0 to enable the counters
10	FLUWI0	Flush with Interrupt 0	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters and generate an interrupt if at least one frame was transmitted, then write 0 to enable the counters
11:19	FTHR1	Frame Count Threshold 1	Reset value of the field is 0b000000001. Frame count threshold for transmit side of MAL channel 1.

20	FLUSH1	Flush with no Interrupt 1	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters without generating an interrupt, then write 0 to enable the counters
21	FLUWI1	Flush with Interrupt 1	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters and generate an interrupt if at least one frame was transmitted, then write 0 to enable the counters
22:31		Reserved	

25.12.2 Interrupt Coalescing Control Receive Register (SDR0_ICCRRX)

SDR0_ICCRRX access is R/W.

Offset: 0x430C

Figure 25-32. Interrupt Coalescing Control Receive Register (SDR0_ICCRRX)

0:8	FTHR0	Frame Count Threshold 0	Reset value of the field is 0b000000001. Frame count threshold for receive side of MAL channel 0 .
9	FLUSH0	Flush with no Interrupt 0	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters without generating an interrupt, then write 0 to enable the counters
10	FLUWI0	Flush with Interrupt 0	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters and generate an interrupt if at least one frame was transmitted, then write 0 to enable the counters
11:19	FTHR1	Frame Count Threshold 1	Reset value of the field is 0b000000001. Frame count threshold for receive side of MAL channel 1.
20	FLUSH1	Flush with no Interrupt 1	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters without generating an interrupt, then write 0 to enable the counters
21	FLUWI1	Flush with Interrupt 1	Reset value of the field is 0b0. Write 1 to clear the timer and frame counters and generate an interrupt if at least one frame was transmitted, then write 0 to enable the counters
22:31		Reserved	

25.12.3 Interrupt Coalescing Timer Register Transmit Channels 0 and 1 (SDR0_ICTRTXx)

SDR0_ICTRTXx access is R/W.

Offset: 0x430D, 0x430E

Figure 25-33. Interrupt Coalescing Timer Register Transmit Channel x (SDR0_ICTRTXx)

0:31		Transmit Timer Register	Configure the value of the timer counter for transmit side of channel 0 or 1 of the MAL. Setting the value to zero (default) disables the timer.

25.12.4 Interrupt Coalescing Timer Register Receive Channel 0 and 1 (SDR0_ICTRRXx)

SDR0_ICTRRXx access is R/W.

Offset: 0x430F, 0x4310

Figure 25-34. Interrupt Coalescing Timer Register Receive Channel x (SDR0_ICTRRXx)

0:31		Receive Timer Register	Configure the value of the timer counter for receive side of channel 0 or 1 of the MAL. Setting the value to zero (default) disables the timer.

25.12.5 Interrupt Coalescing Status Register Transmit Channel 0 and 1 (SDR0_ICSRTXx)

SDR0_ICSRTXx access is Read/Clear.

Offset: 0x4307, 0x4308

Figure 25-35. Interrupt Coalescing Status Register Transmit Channel x (SDR0_ICSRTXx)

0:31		Transmit Timer Status Register	Read and reset the timer counter of the transmit side of channel 0 or 1 of the MAL.

25.12.6 Interrupt Coalescing Status Register Receive Channel 0 and 1 (SDR0_ICSRRXx)

SDR0_ICSRRXx access is Read/Clear.

Offset: 0x4309, 0x430A

Figure 25-36. Interrupt Coalescing Status Register Receive Channel x (SDR0_ICSRRXx)

0:31		Transmit Timer Status Register	Read and reset the timer counter of the receive side of channel 0 or 1 of the MAL.

26. EMAC to PHY Interface Bridges

The PPC440EPx/GRx provides two bridges called the ZMII and RGMII bridges that connect the Ethernet media access controllers (EMACs) to standard physical devices (PHYs).

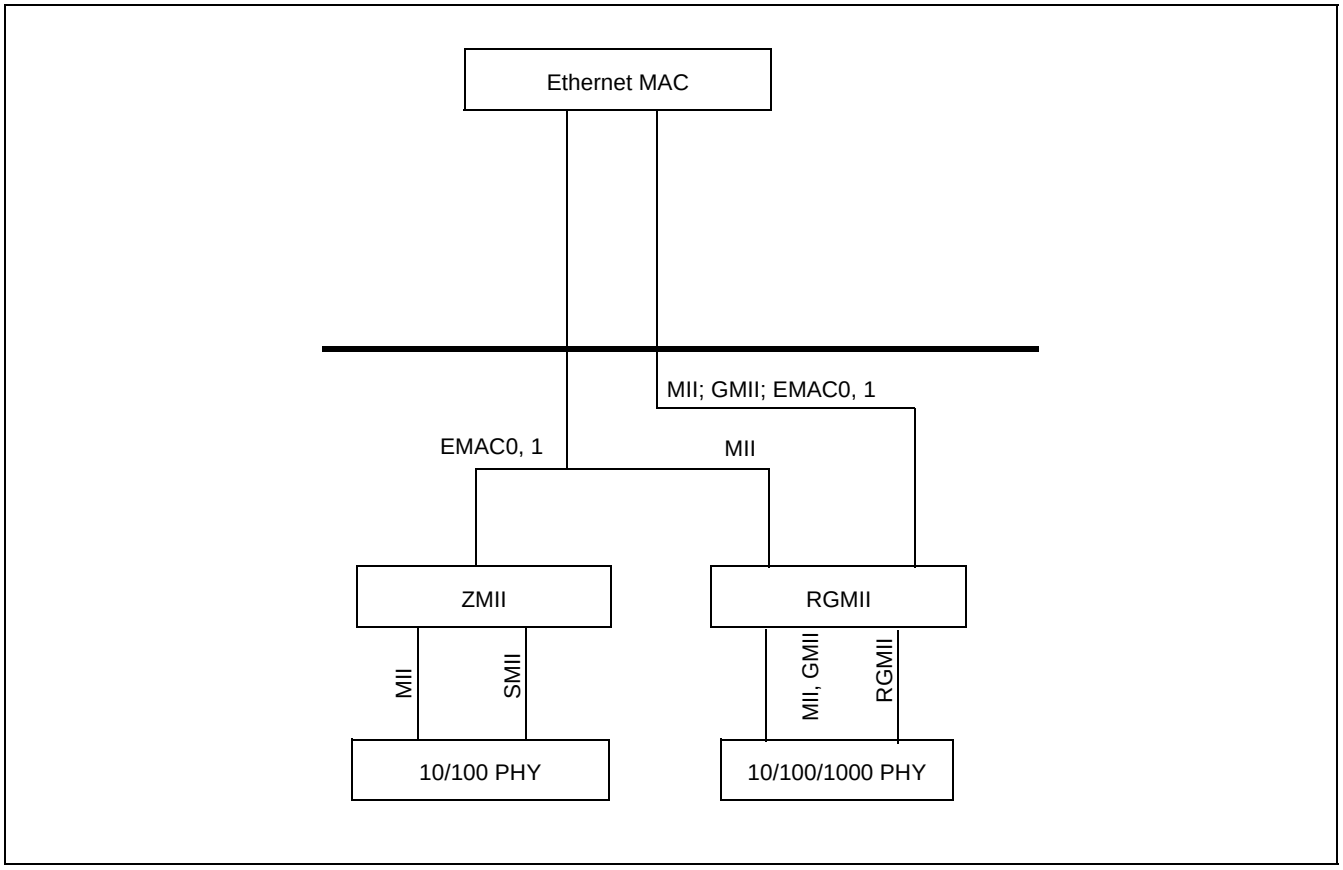
The Ethernet controllers have a variety of interfaces available to communicate with a PHY. For EMAC0 and EMAC1 which support 10/100/1000 Mbps mode, the options are either one MII (media independent interface) or two SMII (serial MII) interfaces. For gigabit (1000 Mbps) speeds one controller may use the full GMII (gigabit MII) or each controller can use RGMII (reduced GMII). *Table 26-1* describes the various Ethernet combinations.

The desired combination is selected by programming the Ethernet group field in the *Pin Function Control Register 1 (SDR0_PFC1)* on page 178 and *Ethernet PLL Configuration Register (SDR0_PFC2)* on page 691. Also, GPIO16:19 and GPIO24:25 are shared with some modes for Ethernet.

Table 26-1. Ethernet Combinations

Option #	EMAC0	EMAC1	SDR0_PFC1[7:9]	SDR0_PFC2[0:2]
0	SMII 0	SMII1	0b010	0b010
1	RGMII 0	RGMII1	0b101	0b101
2	GMII/MII (via RGMII bridge)		0b011	0b011
3	MII (via ZMII bridge)		0b000	0b000

Figure 26-1. ZMII andRGMII Bridge Configurations



26.1 ZMII Bridge

Media independent interface (MII) and management data interface (MDI) signals to and from an EMAC are passed to the ZMII bridge. The ZMII bridge passes MII and MDI signals through to the PHY, or formats the MII signals to support the serial media independent interface (SMII).

Depending on the ZMII configuration, ZMII provides one of the following off-chip interfaces: one MII or two SMII. The interfaces are mutually exclusive. Interface selection is controlled by the Function Enable Register (ZMII0_FER).

SMII significantly reduce the external signal count required to support multiple EMAC ports. The interfaces share two management data interface (MDI) signals (EMCMDIO and EMCMDCLK). While the MII requires 16 additional pins for a single interface, SMII requires only four pins (two data, sync, and a clock). Configuring the ZMII bridge for SMII allows the use of both EMACs while configuring the ZMII bridge for MII allows the use of one EMACs.

ZMII configuration for SMII is done by enabling Pin Function Control Register 1 bits 7:9 (SDR0_PFC1[SELECT] and Ethernet PLL Configuration bits 0:2 (SDR0_PFC2[SELECT]. See *Pin Function Control Register 1 (SDR0_PFC1)* on page 178 and *Ethernet PLL Configuration Register (SDR0_PFC2)* on page 691 for additional information.

26.2 ZMII Features

The ZMII bridge features:

- Support for one MII PHY
- Support for two SMII PHYs
  - 10Mbps and 100Mbps data rates
  - Half and full duplex operation
  - System clock sharing for multiple EMACs and PHYs
  - Independent 1-bit transmit and receive data paths
- Programmable selection of any EMAC to drive MDI
- Programmable selection of EMAC0 to drive MII
- Programmable selection of EMAC0 and EMAC1 in SMII mode 2.

26.3 ZMII Bridge Interface Signals

Each chip pin (ball) is associated with one or more ZMII bridge signals. The ZMII bridge signal associated with each pin depends on the ZMII bridge interface selection. Please refer to the *Ethernet Interface* section of the *Signal Functional Description* table in the PPC440EPx and PPC440GRx *Embedded Processor Data Sheets* for a complete list of all of these signals, how they are grouped, and their I/O characteristics.

26.4 EMAC-ZMII Bridge Interfaces

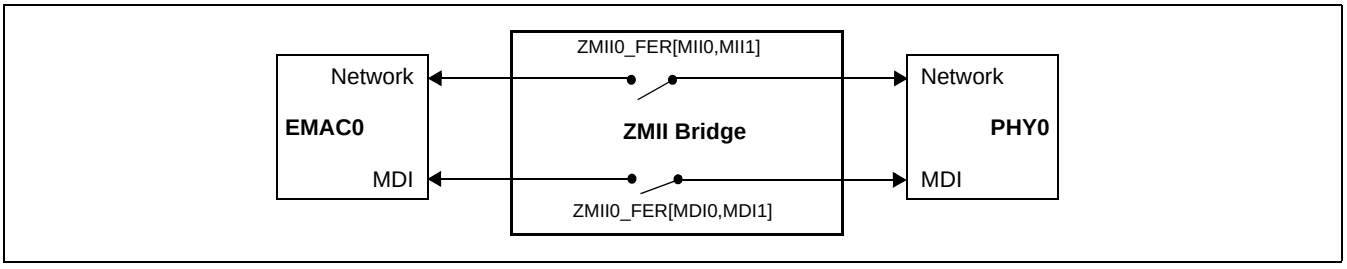
The ZMII bridge attaches to the EMACs on the standard MII, which consists of three sections: transmit data, receive data, and MDI. The transmit and receive data sections move data nibbles across the interface at 25 MHz and 2.5 MHz for 100 Mbps or 10 Mbps operation, respectively. Control signals define valid data, error conditions, and collision detection.

The standard MDI interface enables software to access registers in the PHY and consists of a single bidirectional data signal and a clock driven from ZMII. The MDI connection is always a pass-through interface in the ZMII Bridge.

26.4.1 MII Interface

When the ZMII bridge is configured for MII, the MII signals from one EMAC are simply passed through the ZMII bridge to a PHY. *Figure 26-2* shows an EMAC connected to the ZMII bridge and a PHY. If using MII mode, only a single EMAC may be used.

Figure 26-2. EMAC to PHY Using MII



26.4.1.1 SMII Interface

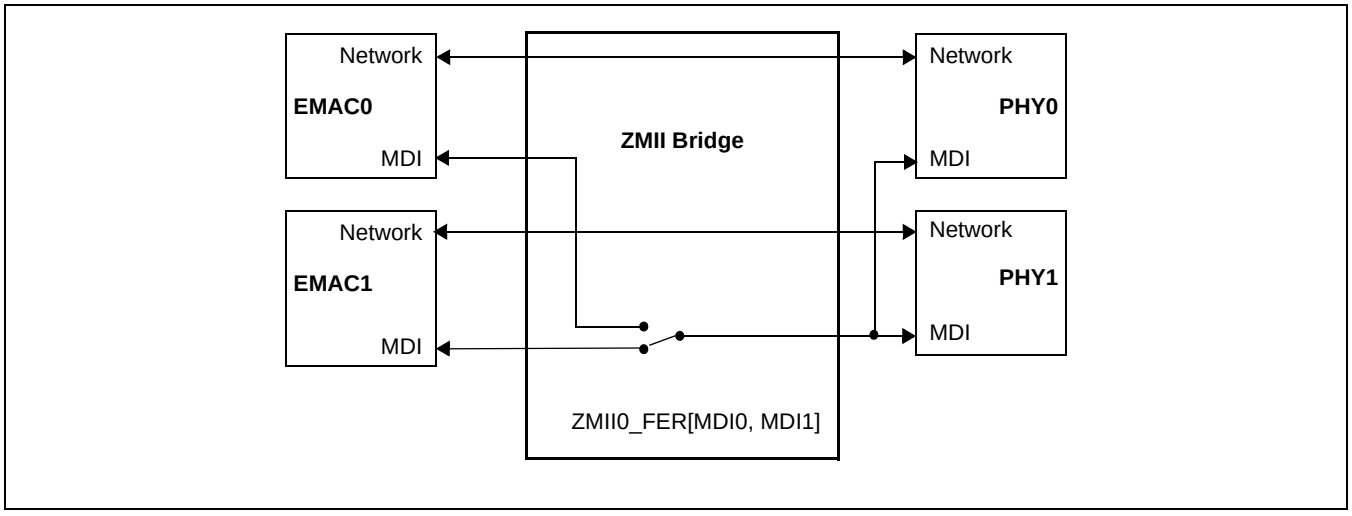
When the ZMII bridge is configured for SMII, MII signal requirements are further reduced by quartering the data width (to a single bit) and multiplying the clock frequency by 10. All signals are synchronous to a 125 MHz clock and the sync signal.

Receive information (data and control) is moved in 10-bit segments. At 100 Mbps, each segment represents a new byte. At 10 Mbps, each segment is repeated ten times. Segment boundaries are defined by a sync signal. The ZMII bridge generates a pulse on the sync signal every 10 clocks.

Transmit operates in the same way. Transmit information (data and control) is sent in 10-bit segments. At 100 Mbps, a new byte is sent every 10 clocks. At 10 Mbps, each segment is repeated 10 times; the PHY can sample any of the repeated 10-bit segments.

Figure 26-3 shows the connection of the two EMACs to the ZMII bridge with the corresponding two PHYs.

Figure 26-3. EMAC to PHY Using SMII



26.5 ZMII Bridge Registers

This section describes the registers in the ZMII bridge, which are listed in *Table 26.2*. The ZMII bridge registers are accessed using an OPB slave interface.

Table 26-2. Register Address List

Register	Description	Address	Access	Page
ZMII0_FER	ZMII Function Enable Register	0x1 EF60 0D00	R/W	685
ZMII0_SSR	ZMII Speed Select Register	0x1 EF60 0D04	R/W	685
ZMII0_SMIISR	ZMII SMII Status Register	0x1 EF60 0D08	R/W	686

26.5.1 Function Enable Register (ZMII0_FER)

ZMII_FER selects the various interface conversion functions provided by the ZMII bridge. ZMII_FER also deselects (functionally isolates) an unused EMAC. The interface conversion functions are mutually exclusive, that is, all EMACs must convert to the same interface (SMII or MII) in the bridge, or be deselected (an EMAC must be deselected for MII). Mixed interface conversion is not supported.

Figure 26-4. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if MII1 is 1.
2		Reserved	
3	MII0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if MII0 or SMII1,or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if MII0 or MII1 1.
6		Reserved	
7	MII1	EMAC1 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, SMII1, or MII0 is 1.
8:31		Reserved	

26.5.2 Speed Select Register (ZMII0_SSR)

ZMII_SSR selects the speed at which each EMAC transfer data (10 Mbps or 100 Mbps). When 10 Mbps is selected, the ZMII bridge generates a 2.5 MHz clock. When 100 Mbps is selected, the ZMII bridge generates a 25 MHz clock.

ZMII_SSR also controls whether collisions are indicated.

Figure 26-5. Speed Selection Register (ZMII0_SSR)

0		Reserved	
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is asserted. 1 EMAC0 collision indication signal is deasserted.	

2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SSR[SP0] does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is asserted. 1 EMAC1 collision indication signal is deasserted.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SSR[SP1] does not override IPG status field. 1 ZMII0_SSR[SP1] overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Can be programmed for SMII; ignored for MII.
8:31		Reserved	

26.5.3 SMII Status Register (ZMII0_SMIISR)

ZMII_SMIISR contains the status transferred during the last inter-packet gap in the SMII interface, at the moment it is read. This register is used only if SMII interfaces are enabled. The register bits pertain to EMAC0RxD and EMAC1RxD SMII inter frame status used to convey packet data, RX_ER, and PHY status and are valid only when RX_DV (receive data valid) is set to 0.

Figure 26-6. SMII Status Register (ZMII0_SMIISR)

0	E01	EMAC0RxD Set to 1	
1	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
2	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
3	E0J	EMAC0RxD Jabber 0 OK 1 Error	
4	E0L	EMAC0RxD Link 0 Down 1 Up	
5	E0D	EMAC0RxD Duplex 0 Half 1 Full	
6	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	

7	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
8	E11	EMAC1RxD Set to 1	
9	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
10	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
11	E1J	EMAC1RxD Jabber 0 OK 1 Error	
12	E1L	EMAC1RxD Link 0 Down 1 Up	
13	E1D	EMAC1RxD Duplex 0 Half 1 Full	
14	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	
15	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
16:31		Reserved	

26.6 RGMII Bridge

The RGMII bridge provides the capability for the EMACs to connect to external Ethernet PHYs that support the reduced gigabit media independent interface (RGMII) or GMII. A GMII port requires twenty four pins of data and control clocks while the RGMII port requires twelve pins plus a common clock.

26.6.1 RGMII Bridge Features

The RGMII bridge supports the following features:

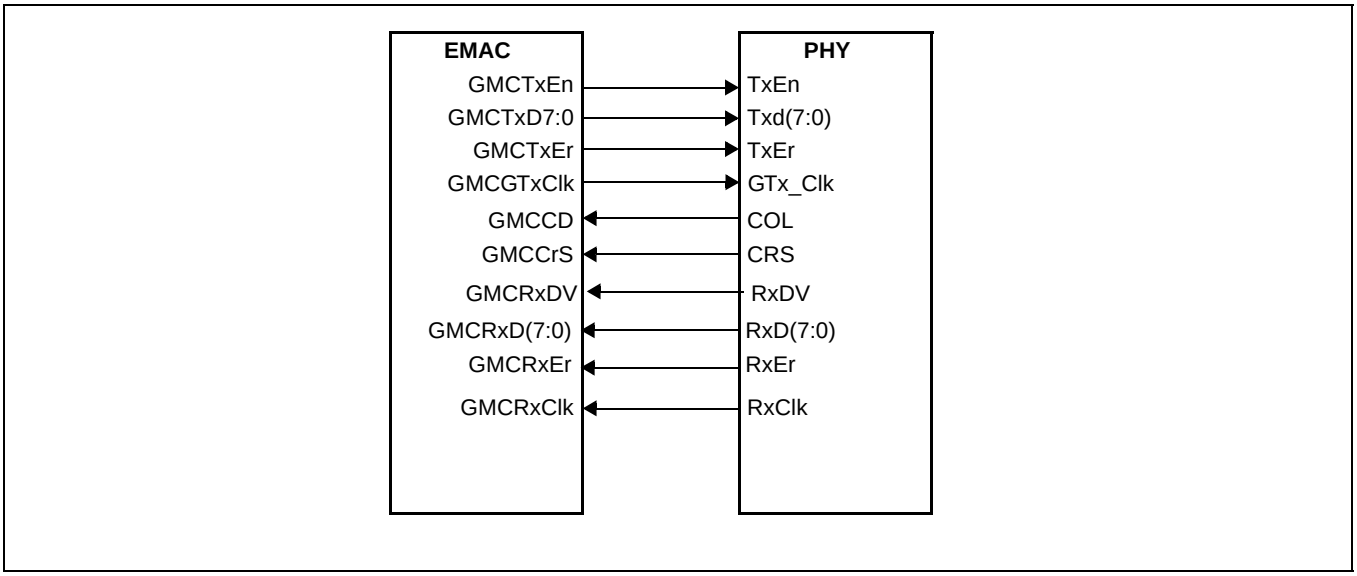
- One GMII PHY
- Two independent RGMII PHYs
- One RGMII PHY
- 10 Mbps, 100 Mbps or 1Gbps data rates
- Uses single 125Mhz clock reference sourced from an external source
- Independent 4-bit transmit and receive paths
- Complies with 2.5V CMOS interface voltages as defined by JEDED EIA/JESD8-5

26.6.2 RGMII Bridge Interface

The purpose of RGMII bridge is to reduce the pin count of the Ethernet gigabit medium independent interface (GMII) by multiplexing the data and control signals.

In the normal GMII mode (as shown in *Figure 26-7*), data between EMAC and PHY is transmitted and received over two eight-bit wide data buses TxD(7:0), RxD(7:0) with six control signals TxEn, TxEr, RxDv, RxEr, COL and CRS. The data and control signals are are synchronous to the two clocks TxClk and RxClk. The total number of pins required by GMII would be 24 plus MDIO and MdClk. (MDIO and MdClk are bidirectional serial data and clock signals used by EMAC to configure the internal registers inside the PHYs.

Figure 26-7. GMII 24 Pins + 2



Using the RGMII Bridge module the databus and control signals between EMAC and PHY are time-multiplexed by using both edges of the clocks as shown in *Table 26-3*.

Table 26-3. RGMII Bridge PHY Interface Signals

RGMII	I/O	Description
GMCnTxClk	I/O	RGMII transmit reference clock is 125MHz, 25Mhx, or 2.5MHz+-50ppm depending on speed
GMCnTxD3:0	O	RGMII transmits bits 3:0 on positive edge of GMCnTxClk and bits 7:4 on negative edge of GMCnTxClk
GMCnTxCtl	O	RGMII transmits GMCTxEn on positive edge of GMCnTxClk and a logical derivative of GMCTxEn and GMCTxEr on negative edge of GMCnTxClk
GMCnRxClk	O	RGMII continuous receive reference clock is 125MHz, 25MHz, or 2.5MHz ± 50ppm depending on speed
GMCnRxD3:0	I	RGMII receives bits 3:0 on positive edge of GMCnRxClk and bits 7:4 on negative edge of GMCnRxClk
GMCnRxCtl	I	RGMII receives GMCRxEn on positive edge of GMCnRxClk and a logical derivative of GMCnRxEn and GMCRxER on negative edge of GMCnRxClk
The data rate achievable at each frequency is as follows: • GMCnTxClk = GMCnRxClk = 125MHz => (125MHz x 2) x 4 bits = 1000Mbits/sec • GMCnTxClk = GMCnRxClk = 25MHz => (25MHz) x 4 bits = 100Mbits/sec) • GMCnTxClk = GMCnRxClk = 2.5MHz => (2.5MHz) x 4 bits = 10Mbits/sec		

26.6.3 Ethernet Signal Grouping for Ethernet to PHY Bridges

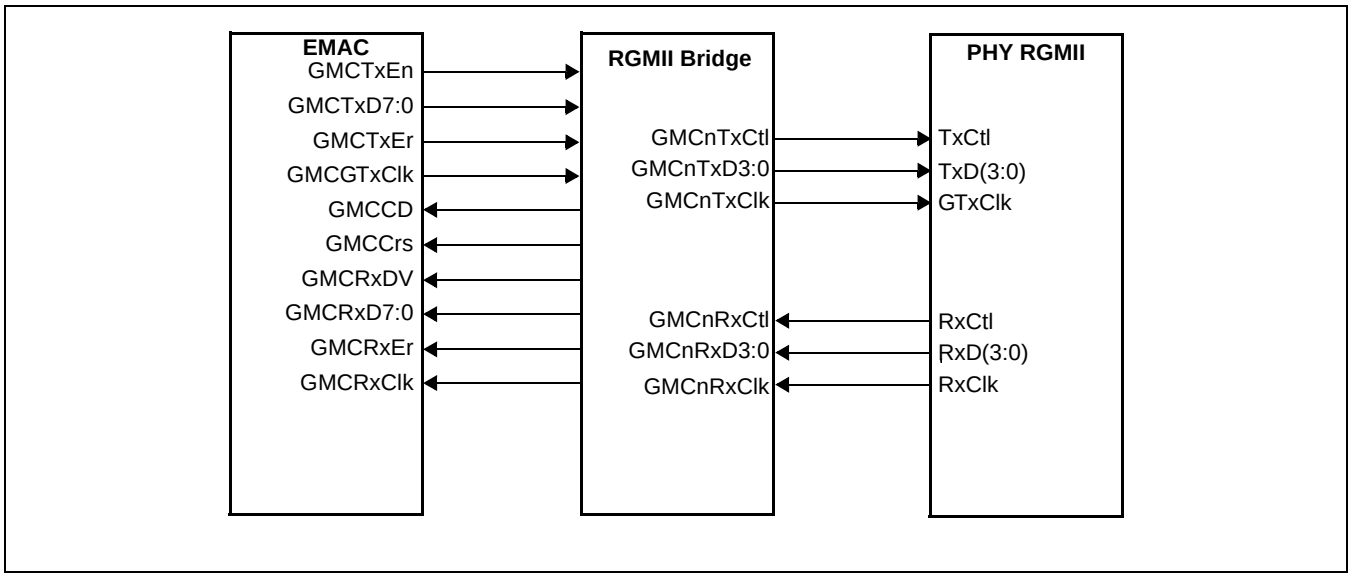
The way a particular signal is used in the Ethernet logic is dependent on the Ethernet mode (MII, GMII, SMII and so on) that the logic is supporting. Please refer to the *Ethernet Interface* section of the *Signal Functional Description* table in the PPC440EPx and PPC440GRx *Embedded Processor Data Sheets* for a complete list of all of these signals, how they are grouped, and their I/O characteristics.

Note: RGMII/GMII does not have the MDIO interface. This interface is driven from ZMII bridge. When using the RGMII/GMII interface, the MDIO interface must be accessed through the ZMII bridge.

26.6.4 EMAC-RGMII Bridge Interfaces

The GMII interface is multiplexed by the RGMII module, which reduces pins by using both edges of clocks (reducing data buses TxD, RxD by half), multiplexing control pins TxEn/TxEr, RxDv/RxEr on two clock edges, and regenerating indicator signals COL, CRS from data stream bit patterns.

Figure 26-8. EMAC to PHY Using RGMII Bridge



26.6.5 RGMII Bridge Registers

This section describes the registers in the RGMII bridge, which are listed in *Table 26-4*. The RGMII bridge registers are accessed using an OPB slave interface.

Table 26-4. RGMII Bridge Registers

Register	Description	Address	Access	Page
RGMII0_FER	RGMII Function Enable Register	0x1 EF601000	R/W	690
RGMII0_SSR	RGMII Speed Select Register	0x1 EF601004	R/W	690

26.6.5.1 RGMII Function Enable Register (RGMII0_FER)

The RGMII function enable register (RGMII0_FER) selects the various interface conversion functions provided by the RGMII bridge. RGMII0_FER also deselects (functionally isolates) an unused EMAC.

Figure 26-9. RGMII Function Enable Register (RGMII0_FER)

0:11		Reserved	
12:13	MDIOEN	MDIO enable Bit 12 on - Select EMAC 0 Bit 13 on - Select EMAC 1	
14:24		Reserved	
25:27	CH1CFG	Channel 1 configuration 0xx Disable 100 Reserved 101 RGMII1 110 Reserved 111 Reserved	
28		Reserved	
29:31	CH0CFG	Channel 0 configuration 0xx Disable 100 Reserved 101 RGMII0 110 Reserved 111 GMII (RGMII Bridge Pass-through)	

Note: In reduced mode, the two channels operate independently. In pass-thru mode, only one channel operates. If both channels are enabled and set to pass-through, or one channel is reduced and the other is pass-through, these are invalid settings resulting in both channels being disabled.

26.6.5.2 Speed Selection Register (RGMII0_SSR)

The speed selection register RGMII0_SSR selects the speed at which each EMAC transfer data (10 Mbps, 100 Mbps or 1000 Mbps). When 10 Mbps is selected, the RGMII bridge generates a 2.5 MHz clock. When 100 Mbps is selected, the RGMII bridge generates a 25 MHz clock. When 1000 Mbps is selected, the RGMII bridge generates a 125 MHz clock. RGMII0_SSR also controls whether collisions are indicated.

Figure 26-10. Speed Selection Register (RGMII0_SSR)

0:20		Reserved	See Table 26-1, page 681
21:23	SP1	EMAC 1 Speed Selection 000 10 Mbps selected. 010 100 Mbps selected. 100 1000 Mbps selected	
24:28		Reserved	
29:31	SP0	EMAC 0 Speed Selection 000 10 Mbps selected. 010 100 Mbps selected. 100 1000 Mbps selected	

26.7 Phase-Locked Loop (PLL) Dedicated to the RGMII Bridge

The RGMII PLL is dedicated to the RGMII bridge. The objective is to guarantee a reference clock to the RGMII bridge with a 50/50 duty cycle at 125MHz with an input reference clock of 125MHz. The RGMII PLL is configured using the following register.

26.7.1 Ethernet PLL Configuration Register (SDR0_PFC2)

Figure 26-11. Ethernet PLL Configuration Register (SDR0_PFC2)

0:2	SELECT	Ethernet pin select EMAC 1	See *Table 26-1 Ethernet Combinations* on page 681
3:6	RANGEA	PLLOUTA frequency selector	Reset value is 0b1001. Recommended value: 0b0011
7:9	RANGEB	PLLOUTB/C frequency selector	Reset value is 0b001. Recommended value: 0b001
10:14	MULTI	Frequency multiplication selector	Reset value is 0b00001. Recommended value: 0b00001
15:24	TUNE	Loop stability tuning bits	Reset value is 0b0100110110 Recommended value: 0b0110110110
25:31		Reserved	
The reset values indicated for each field will ensure proper operation of the bridge.			

27. Ethernet Media Access Controllers

The PPC440EPx/GRx provides two Ethernet media access controllers (EMACs) that are generic implementations of the Ethernet Media Access Control (MAC) protocol complying with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. Both EMACs support half-duplex (CSMA/CD),full-duplex operation for 10-Mbps and 100-Mbps operations, and operation in the 1000 Mbps range (Gigabit Ethernet) complying with IEEE 802.3z. In the Gigabit Ethernet mode, the real bandwidth is limited only by system latency (attributable to memory access layer (MAL), memory, arbitration, etc.).

Both EMACs are implemented identically, with the exception of register addresses. EMAC0 and EMAC1 support 10-Mbps, 100-Mbps, and1000-Mbps operations. Because both EMACs operate identically, the rest of this chapter describes a single EMAC. The EMACs are referred to as EMAC0 and EMAC1. Except in the register summary tables in *EMAC Registers* on page 720, the EMAC registers are prefixed "EMACx_" to denote the identical implementation of registers in each EMAC.

Each EMAC provides one on-chip peripheral bus (OPB based 32-bit bidirectional) slave interfaces and two other extended OPB (EOPB based 128-bit bidirectional) slave interfaces. The first OPB interface provides access to the EMAC configuration and status registers. The PLB/OPB bridge enables the processor core to access these registers.

The two EOPB slave interfaces are used to exchange packet information with the memory access layer (MAL). The MAL is a multi-channel, intermediate hardware layer that resides between packet-based communication cores (such as EMAC) and external memory (such as SDRAM or SRAM). The MAL transfers packet information and status between the EMACs and external memory separately for each of the two EMAC channels (one receive and one transmit). Software (a device driver) maintains a buffer descriptor ring and a set of data buffers in external memory for each channel, and manages the exchange of packet data between the data buffers and the software protocol.

The MAL performs functions such as arbitration between service requests, handling the buffer descriptor memory structure, updating the descriptor status/control fields at the end of packet transfer, and so on. EMAC supports up to 64 words burst length transactions on the MAL interface (which can be programmed to 128 words burst).

In 10/100 Mbps range each EMAC uses a media independent interface (MII) to communicate with the Z media independent interface (ZMII) bridge. The ZMII bridge, described in detail in *EMAC to PHY Interface Bridges* on page 681 communicates with standard physical devices (PHYs). The ZMII bridge supports the media independent interface (MII) and serial media independent interface (SMII). Details of PHY communication are in *EMAC-ZMII Bridge Interfaces* on page 683.

EMAC uses independent receive and transmit FIFOs. Programmable FIFO thresholds minimize overflows and underruns, and can launch integrated IEEE 802.3x pause packets for flow control.

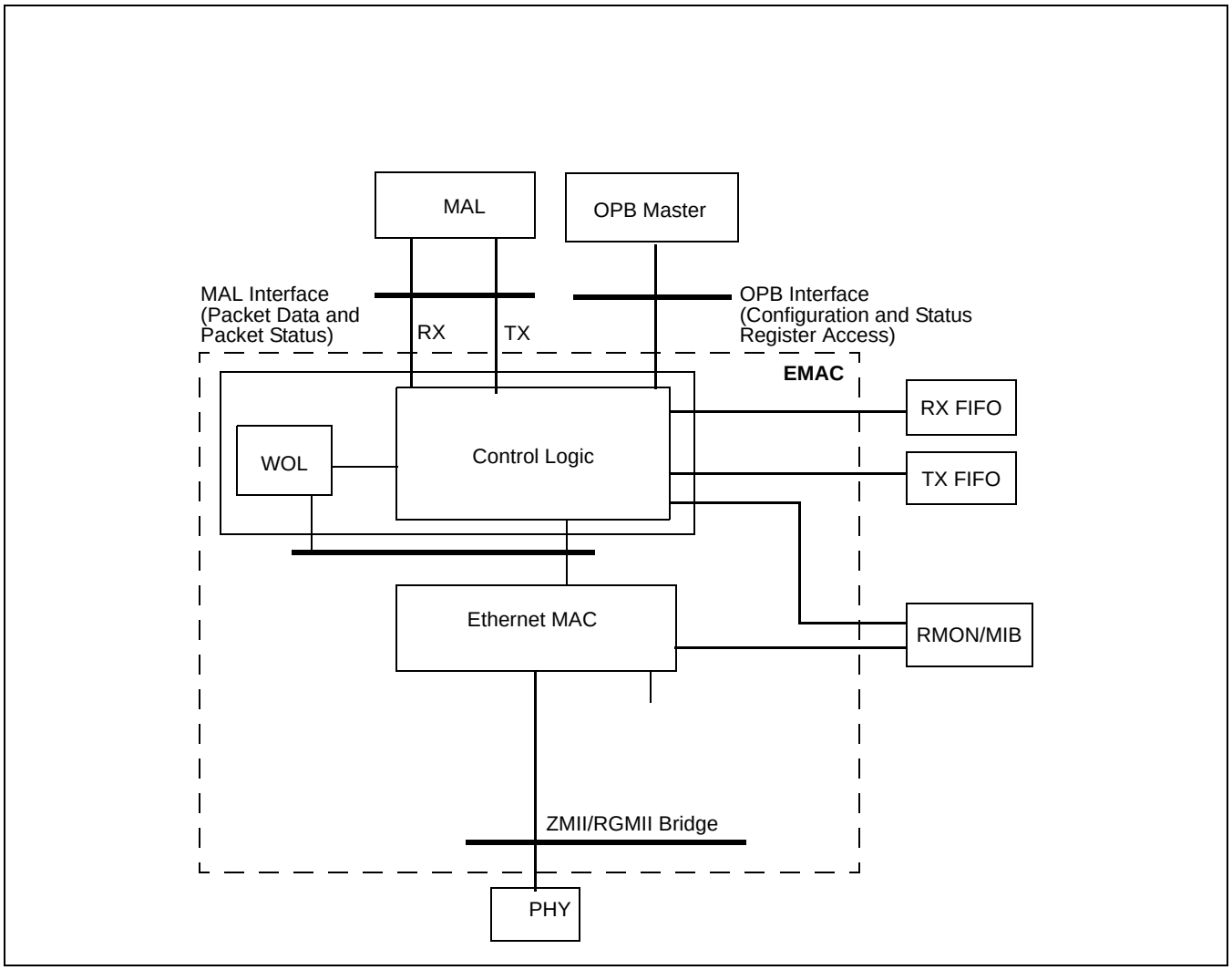
As part of the remote monitoring (RMON) and management information base (MIB) defined in IEEE 802.3z, the EMAC contains registers that count the number of octets transmitted and received.

EMAC supports jumbo packets with no VLAN tag and with VLAN tag, for up to 9018 and 9022 bytes respectively. To enable jumbo packet support, the jumbo enable bit 20 of EMAC mode register 1 must be set as follows: EMACx_MR1[JPSM=1]. In addition to enabling the EMAC to transmit and receive jumbo packets, the MAL driver must support packets received by multiple descriptor/buffers.The MAL, implemented in the PPC440EPx/GRx, supports a maximum of 4080 bytes.

In order to receive packets greater than the maximum buffer size, a packet greater than 4080 would have to be transmitted or received using multiple descriptors. By using the first and last bits in the descriptor field, software is able to determine the beginning, middle, and end of a packet. For example, if a 9K packet is received and the receive buffer size (RCBSx) is set to the maximum - 0x000000FF, the first descriptor will have its first bit set, and

the second will not have either the first or last bit set, and the third descriptor will have the last bit set. The device driver will be required to move these multiple buffers into contiguous memory before passing the packet to the IP stack or use an interface that supports sending a link list of buffers to the IP stack.

Figure 27-1. EMAC in a Typical Ethernet Application



Note: Each EMAC is connected to an OPB, to which the OPB master and MAL are also connected.The EMAC transmit channel operates independently of the receive channel.

27.1 EMAC Features

EMAC features:

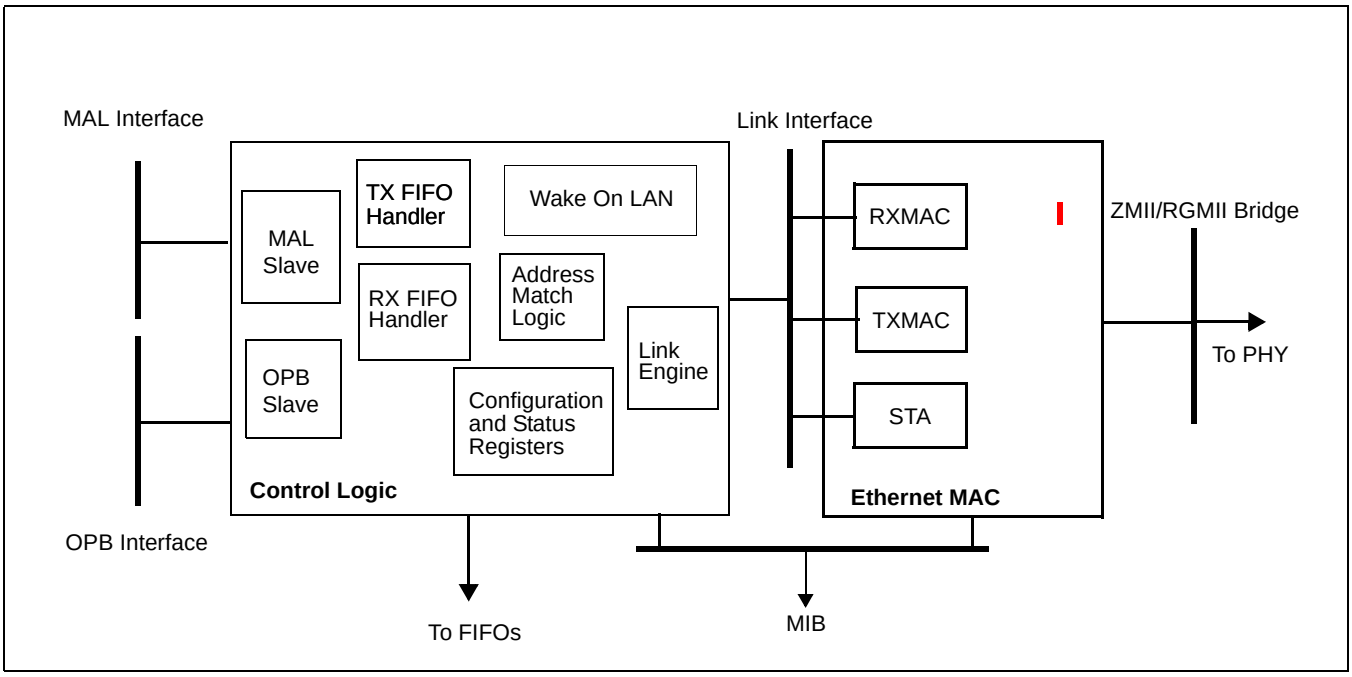
- Triple speed (10/100/1000 Mbps) CSMA/CD (half-duplex) and full-duplex Ethernet MAC complying with ANSI/IEEE Std. 802.3, 802.3z and IEEE 802.3u supplement.
- Two 128-bit extended OPB (EOPB) interfaces to MAL with one receive and one transmit channel enable high throughput and performance which allows the interfaces to be connected to different MALs in one system via the same or separate MAL-EOPB buses.
- EMAC TX channel has a multiple status mechanism which allows EMAC to send the status/error word that is relevant to the last TX packet only after the next TX packet.
- Support for next page feature in the PCS auto-negotiation process.
- Automatic source address insertion or replacement for transmitted packets is a programmable option.
- Automatic stripping of frame padding bytes and frame check sequence (FCS) is a programmable option.

  When padding bytes are stripped, the padding and FCS field are removed. FCS stripping removes only the FCS field.
- FCS control for transmit/receive packets.
- Access to registers with support for burst processing.
- MAL for packet moving having one-cycle MAL slave latency.
- Independent, large (2 KB) transmit and up to (16 KB) receive FIFOs with programmable thresholds to minimize overruns and underruns and parity bits indication to the FIFOs and LPRA entries.
- Multiple packet handling in transmit and receive FIFOs.
- Unicast, multicast, broadcast, and promiscuous address filtering capabilities.
- Two 64-bit hash filters for unicast and multicast packets.
- Automatic retransmission of collided packets.
- Rejection of runt packets before providing them to MAL.
- MII for connection to a variety of PHY layer devices (used when 10/100 Mbps mode is chosen).
- Progammable interface for connecting to a Gigabit PHY. A gigabit media independent interface (GMII) or a PMA interface can be used for connection to the attached PHY layer device (used when 1000 Mbps mode is chosen).
- Programmable inter-packet gap to enable tuning for better system performance.
- Compliance with IEEE 802.3x standard packet-based flow control, including self-assembled control pause packet transmitting.
- Support for VLAN tag ID in compliance with IEEE Draft 802.3ac/D1.0 standard.
- VLAN tag insertion or replacement for transmit packets is a programmable option.
- Wake on LAN (WOL) handling.
- Programmable internal and external loop-back capabilities.
- Extensive error/status vector generation for each processed packet.
- Power management using a clock and power management (CPM) unit.
- Internal PCS capabilities with an internal PCS for use in the 1000 Mbps with a PHY PMA interface

27.2 EMAC Operation

The EMAC hardware components and its internal structure are illustrated in *Figure 27-2*. The EMAC is connected to the physical layer device via the MII (for 10/100 Mbps Ethernet), the GMII ZMII/RGMII bridge. The active PHY interface (MII, GMII or PMA) is defined via the EMAC configuration.

Figure 27-2. Internal EMAC Structure



The control logic sub-block implements the following functions:

- OPB slave device
- MAL slave device
- FIFO management logic
- Ethernet address and pause packet match logic (address match logic)
- Register file for FIFOs and Ethernet MAC handler management
- Logic for support of WOL technology
- Link engine

The Ethernet MAC sub-block implements the following functions:

- Transmit MAC Handler (TXMAC)
- Receive MAC Handler (RXMAC)
- MII/GMII management function unit (STA)

The above functions are described in the following sections.

27.2.1 MAL Slave Logic

The MAL slave (MALS) logic controls MAL transactions. MALS transfers TX and RX data between MAL and the OPB on one side, and the EMAC FIFO handlers on the other side. MALS is a dedicated MAL slave.

27.2.2 OPB Slave Logic

The OPB slave (OPBS) logic controls all OPB transactions between the processor core and the EMAC configuration and status registers.

27.2.3 FIFO Management Logic

The FIFO management logic is used for data interchange handling with the attached receive (RX FIFO) and transmit (TX FIFO modules.

27.2.4 Ethernet Address Match Logic

Address match logic checks the destination address of received packets against a set of predefined addresses specified by the current address filtering mode. EMAC contains one unicast (individual address) register, two hash tables for filtering individual and group address, and logic for detecting broadcast address (all ones). EMAC supports promiscuous mode and multicast promiscuous mode.

This logic also checks the destination address of the incoming packet against a special multicast address used for control (pause) packet recognition.

All checks for address matching are performed only after the entire destination address field is received (except for promiscuous and multicast promiscuous modes).

27.2.5 Configuration and Status Registers

Configuration and status registers define the EMAC configuration and reflect error/status of recent transmitted or received packets.

27.2.6 Wake On LAN Logic

EMAC supports Wake On LAN (WOL) technology, an industry standard described in the *Wired for Management (WFM)* specification. This technology allows a sleeping or powered-off network node to be awakened with a special packet called a Magic Packet. In the PPC440EPx/GRx, with WOL mode enabled, the EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1).

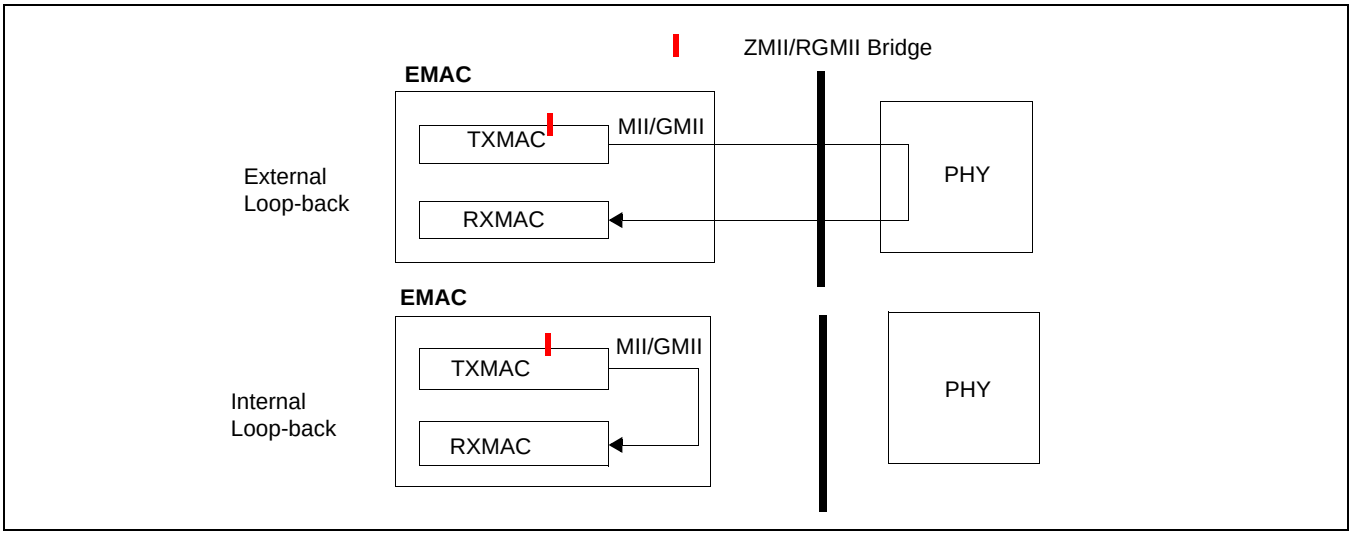
27.2.7 Ethernet MAC

The EMAC gigabit media independent interface (GMII/MII) connects to an external PHY through the ZMII/RGMII bridges. The ZMII/RGMII bridges enables the EMAC to support GMII, MII, RGMII and SMII PHY.

27.2.8 EMAC Loopback Modes

EMAC supports the external and internal loop-back modes illustrated in *Figure 27-3*.

Figure 27-3. EMAC Loopback Modes



External loop-back mode uses the PHY device. To EMAC, external loop-back is identical to full-duplex operation. Configuring EMAC to operate in a full-duplex mode enables external loop-back. EMAC does not need an external loopback configuration signal.

In internal loopback mode, data from the EMAC transmit channel is routed to the EMAC receive channel. The loop-back mode functions correctly with or without a connected PHY. In internal loopback mode, EMAC does not activate (monitor) any MII/GMII signals. Transmit channel signals are buffered internally to the receive channel. However, if internal loopback is used without a PHY, the EMAC transmit and receive clocks must be provided by another means (enabling ethernet clock select bit (SDR0_MFR[ECS] when loopback is used without a PHY. See *Miscellaneous Function Register (SDR0_MFR)* on page 181). In internal loopback mode, the EMAC transmit clock and receive clock must be sourced from a single clock. Also in internal loop-back, the internal loop-back enable bit 1 in Mode Register 1 is set.

27.2.9 Packet Rejection

EMAC has a packet reject input signal, RejectPkt, that can be used by external logic to identify whether the current received packet should be accept or rejected. When RejectPkt is a logic 0, the current receive packet is sent to the MAL. When the RejectPkt is a logic 1, the current received packet is removed from the Receive FIFO.

The polarity of the RejectPkt0:1 signals is programmable by SDR0_MFR[PRP].

The RejectPktn signal is enabled for EMAC0 and EMAC1 by SDR0_MFR[PRE0, PRE1].

Ther number of rejected packets for EMAC0 and EMAC1 are recorded by the EMAC Packet Reject Counters, SDR0_EMAC0REJCNT and SDR0_EMAC1REJCNT.

Table 27-1. EMAC Status and Packet Reject Counter Registers

Register	Offset	Access	Description
SDR0_EMAC0RXST	0x4301	R/W	EMAC0 RX Status Register

Table 27-1. EMAC Status and Packet Reject Counter Registers

Register	Offset	Access	Description
SDR0_EMAC0TXST	0x4302	R/W	EMAC0 TX Status Register
SDR0_EMAC1RXST	0x4303	R/W	EMAC1 RX Status Register
SDR0_EMAC1TXST	0x4304	R/W	EMAC1 TX Status Register
SDR0_EMAC0REJCNT	0x4305	R	EMAC0 RX Packet Reject Counter
SDR0_EMAC1REJCNT	0x4306	R	EMAC1 RX Packet Reject Counter

27.2.9.1 EMACx RX Status Register

Figure 27-4. EMACx RX Status Register (SDR0_EMACxRXST)

0:1		Reserved	
2	FOR	RXFIFOOverrun	The current frame was aborted because an overrun. Flow of received data applied to the FIFO was detected, and the data was invalidated since there was no empty space in the Receive FIFO.
3	BC	BroadcastAddr	The frame was received with the broadcast destination address and at least 6 bytes are received.
4	MC	MulticastAddr	The frame was received with a group destination address other than broadcast and at least 6 bytes are received.
5	UC	UnicastAddr	The frame was received with a unicast destination address and at least 6 bytes are received.
6:8	UPR	User Priority Field	Provides the content of User Priority field (bits 7:5 respectively) from received VLAN Tagged frame.
9	VLAN	RX VLAN Tagged Frame	The currently received frame was a VLAN Tagged frame.
10	LOOP	Received in Loop-Back Mode	The currently received frame is an echo of frame transmitted by TXMAC block.
11	UOP	Rx Unsupported Opcode	The currently received frame was a control frame with an unsupported opcode.
12	CPF	RX Control Pause Frame	The currently received frame was a pause frame.
13	CF	RX Control Frame	The currently received frame was a control frame.
14	MSIZ	1024–MaxSize bytes received	Indicates that the currently received frame (including bad frames) has a byte length between 1024 and maxFrameSize. This is defined by the Jumbo Packets support enabled and VLAN support enabled bits as follows:
15	1023	512–1023 bytes received	The currently received frame (including bad frame) was between 512 and 1023 bytes length (excluding framing bits but including FCS bytes).

Jumbo Packets Support Enabled	VLAN Support Enabled	maxFrameSize (Bytes)
0	0	1518
0	1	1522
1	0	9018
1	1	9022

16	511	256–511 bytes received	The currently received frame (including bad frame) was between 256 and 511 bytes length (excluding framing bits but including FCS bytes).
17	255	128–255 bytes received	The currently received frame (including bad frame) was between 128 and 255 bytes length (excluding framing bits but including FCS bytes).
18	127	65–127 bytes received	The currently received frame (including bad frame) was between 65 and 127 bytes in length (excluding framing bits but including FCS bytes).
19	64	64 bytes received	The currently received frame (including bad frame) was 64 bytes in length (excluding framing bits but including FCS bytes).
20	RUNT	Runt frame	The frame length was less than MinFrameSize (64 bytes, excluding the preamble and SFD bytes) and a Short Event was not detected.
21	SEVT	Short event	The duration of PHY_RX_DV signal was less than the Short-EventMaxTime constant. For this indication, GMAC always assumes that the preamble and SFD fields contain eight bytes regardless of the number actually received.
22	AERR	Alignment error	The received frame is not an integral number of octets in length. This is applicable only to 10/100 Mbps operation.
23	SERR	Received with symbol error	For 100 Mbps operation, indicates a case when a valid carrier was present and there was at least one occurrence of Data Reception Error. For 1000 Mbps operation, indicates a case when the receiving media is non-idle (a carrier event) for a period of time greater then or equal to slotTime for half duplex, or greater then or equal to minFrameSize for full-duplex, and during which there was at least one occurrence of an event that causes the PHY to indicate Data reception error on the GMII.
24		Reserved	
25	BURST	Received burst	Indicates that carrier event (from the first byte of DA until the end of transmission) was equal or greater than slotTime. This is applicable only in HDX mode at 1000 Mbps.
26	F2L	Frame is too long	The length of the received frame exceeded the maximum allowed value (maxFrameSize): - 1518 octets for standard frame (checked only when the length/type field of the transmitted frame contains a length value and REG_JUMBO_EN is inactive) - 1522 octets for VLAN tagged frame (checked only when the length/type field of the transmitted frame contains a length value and REG_JUMBO_EN is inactive) - 9018 octets for a jumbo frame (with no VLAN support) - 9022 octets for a VLAN tagged jumbo frame
27	OERR	Out of range length error	The received frame has a length field value greater than the maximum allowed LLC data size (greater than 1500 and less than 1536).
28	IERR	In range length error	The content the length field in the received frame is less or equal to the maximum allowed MAC Client data size (1500 bytes).
29	LOST	Frame lost due to internal EMAC receive error	Frame reception was interrupted internally within the EMAC.
30	BFCS	Bad FCS in the received frame	Received frame had an FCS value which does not match the FCS value calculated by the EMAC.

31	RXOK	Receive OK	None of the following events occurred during a receive operation: - Receive FIFO Handler interrupted receive process - FCS error - Length mismatch error - Too long frames - Alignment errors - Received with symbol error - Short event
Note: If the receive frame is interrupted internally within the EMAC, then all bits except bits 2, 26, and 29 are in an unpredictable state.			

27.2.9.2 EMACx TX Status Register

Figure 27-5. EMACx TX Status Register (SDR0_EMACxTXST)

0:5		Reserved	
6	FUR	TXFIFOUnderrun	The current frame was aborted because an underrun. The Data indication from the FIFO was not valid in time to allow continuous data transmission on the MII/GMII interface.
7	BC	BroadcastAddr	The frame was transmitted to the broadcast destination address.
8	MC	MulticastAddr	The frame was transmitted to a group destination address other than broadcast.
9	UC	UnicastAddr	The frame was transmitted to a unicast destination address.
10	FP	Frame paused by control packet	Indicates that the currently transmitted frame was delayed due to a pause packet received by EMAC.
11	BFCS	Bad FCS in the transmitted frame	The transmitted frame had an FCS which does not match the FCS calculated by EMAC. EMAC asserts this indication only if the frame being transmitting ended without interference (there was no collision or stop request during transmission).
12	CPF	TX control pause frame	The currently transmitted frame was a pause frame. The control packet was self-assembled by EMAC.
13	CF	TX control frame	The currently transmitted frame was a control frame. The control packet was self-assembled by EMAC.
14	MSIZ	1024–maxsize bytes transmitted	The currently transmitted frame was between 1024 and 1518 (1522 in case of VLAN mode) bytes length (excluding framing bits but including FCS bytes).
15	1023	512–1023 bytes transmitted	The currently transmitted frame was between 512 and 1023 bytes length (excluding framing bits but including FCS bytes).
16	511	256–511 bytes transmitted	The currently transmitted frame was between 256 and 511 bytes length (excluding framing bits but including FCS bytes).
17	255	128–255 bytes transmitted	The currently transmitted frame was between 128 and 255 bytes length (excluding framing bits but including FCS bytes).
18	127	65–127 bytes transmitted	The currently transmitted frame was between 65 and 127 bytes in length (excluding framing bits but including FCS bytes).
19	64	64 bytes transmitted	The currently transmitted frame is 64 bytes in length (excluding framing bits but including FCS bytes).

20	SQE	SQE indication	This bit is applicable only for the HDX 10Mbps. Indicates that Signal Quality Error test has failed. Following the end of a successful EMAC transmission (ended with no collision), EMAC expects the PHY to assert the collision signal until the end of the first part of the inter frame-gap (IFG1). If collision was not asserted, EMAC indicates a signal quality error. When the IGNORE_SQE_TEST bit in Mode Register 1 is set, SQE test is not performed.
21	LOC	Loss of carrier sense	During transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all. This is applicable only in HDX mode. Returns 0 in FDX mode.
22:23		Reserved	
24	IERR	EMAC internal error	Transmit was interrupted internally within EMAC.
25	EDF	Excessive deferral	The current frame has been deferred for an excessive period of time. The value of this period in bits is calculated as follows: - For 10 and 100Mbps operation: 2 x (maxFrameSize x 8) bit times - For 1000Mbps operation: 2 x (burstLimit + maxFrameSize x 8 + headerSize) bit times. This is applicable only in half-duplex mode.
26	ECOL	Excessive collisions	The current frame transmission had ended with a collision at the 16th consecutive attempt. This is applicable only in half duplex mode.
27	LCOL	Late collision	The frame collided with other data outside of the collision window. This is applicable only in half duplex mode.
28	DFFR	Deferred frame	The current frame transmission is deferred due to activity on the medium on its first transmission attempt. (Note that if EMAC waits to the end of the IFG period without indicating an active medium, the frame is not considered as a deferred frame). This is applicable only in half duplex mode.
29	MCOL	Multiple collision frame	The transmitted frame collided with other data on the bus more than once, but less than 16 times. This is applicable only in half duplex mode.
30	SCOL	Single collision frame	The transmitted frame collided with other data on the bus. This is applicable only in half-duplex mode.
31	TXOK	Transmit OK	None of the following events was discovered during transmission: - Collision - Late collision - Lost of carrier sense - Excessive deferral - Bad FCS - TX FIFO Handler interrupted transmit process - SQE error (for HDX 10Mbps only)
Note: If the transmit frame is interrupted internally within the EMAC, then all bits except bits 6, 24, 25, 26, and 27 are in an unpredictable state.			

27.2.9.3 EMACx RX Packet Reject Counter

Figure 27-6. EMACx Reject Count Register (SDR0_EMACxREJCNT)

0:31	RC	Reject counters.	Count the number of packets rejected.

27.3 EMAC Transmit Operation

The transmit part of EMAC handles packet transmission from the MAL device to the MII/GMII or PMA interface. At the end of a transmission process, EMAC provides a status/error word which allows monitoring the transmission operation.

27.3.1 Arbitration Between TX Channels

EMAC's TX channel can be configured to work in either of two ways: single packet mode or multiple packet mode.

In single packet mode, EMACx_MR1[TR] = 0. The channel requests one packet from MAL and resets EMACx_TMR0[GNP=1] when it receives a MAL_TX_STATUS_DONE indication. The channel asks for service again only after EMACx_TMR0[GNP = 1] (set by the device driver).

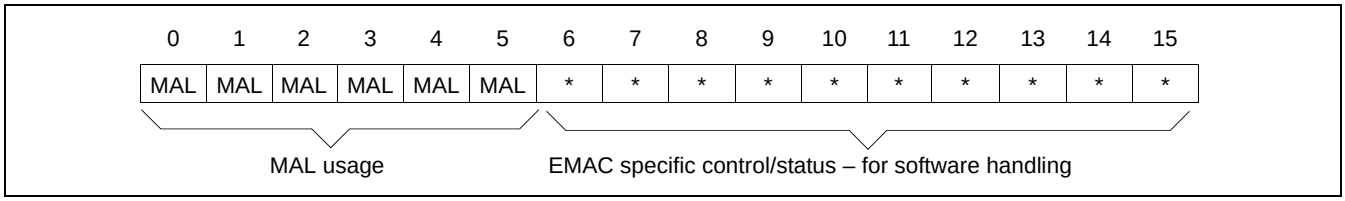
In multiple packet mode, EMACx_MR1[TR] = 1. After the channel finishes transferring a packet and receives a MAL_TX_STATUS_DONE indication, the channel asks MAL for the next packet if there is enough room in the FIFO. The channel continues to request more packets until one of the following events occur:

- The channel receives notification from MAL that the next buffer descriptor is not marked ready for transmission. When this occurs, the channel sets EMACx_TMR0[GNP] = 0, as appropriate, and waits for software to reactivate the channel by setting EMACx_TMR0[GNP] = 1.
- A transmit error or signal quality error (SQE) occurs and the corresponding interrupt is not masked in the EMACx_ISER. After such an error, the channel sets EMACx_TMR0[GNP] = 0, and sets EMACx_ISR[DB] = 1 (the EMACx_ISR field that is set depends on which channel is active) and the corresponding EMACx_ISR error. The channel does not request service again until EMACx_TMR0[GNP] = 1 and EMACx_ISR[DB] = 0.

27.3.1.1 MAL TX Descriptor Control/Status Field

For each transmitted packet, MAL uses the descriptor control/status field of the buffer descriptor to provide an EMAC with control information (write), and to obtain packet status from the EMAC after transmission is complete (read). Software writes the control bits in the buffer descriptor before packet transmission, and reads the status bits from the buffer descriptor after packet transmission has completed. See *Buffer Descriptor* on page 652 for more information on the buffer descriptor structure.

Figure 27-7. MAL TX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See *TX Status/Control Field Format* on page 660	R
TX Control Information (Write Access)			
6	Generate FCS	0 FCS is not generated by EMAC. 1 EMAC calculates and adds the FCS field to the packet to be transmitted.	W

Bits	Bit Name	Bit Description	Mode
7	Generate padding	0 Padding is not generated by EMAC. 1 EMAC adds the padding field to the packet to be transmitted (only when Generate FCS is also set).	W
8	Insert source address	0 EMAC will not insert source address. 1 EMAC inserts the source address field into the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
9	Replace source address	0 EMAC will not replace source address. 1 EMAC replaces the source address field in the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
10	Insert VLAN Tag	0 EMAC will not insert a VLAN tag. 1 EMAC inserts the VLAN Tag field into the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
11	Replace VLAN Tag	0 EMAC will not replace the VLAN tag. 1 EMAC replaces the VLAN Tag field in the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
TX Status Information (Read Access)			
6	Bad FCS on transmitted frame	0 FCS was correct in the transmitted packet. 1 Indicates that a bad FCS was found in the transmitted packet (this bit is also set when parity error bit 5 is set).	R
7	Reserved	Always zero. (However this bit is used by TCP/IP hardware to report TX error.	R
8	Loss of carrier sense	0 No loss of carrier. 1 During the transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all. Applicable only in half duplex mode.	R
9	Excessive deferral	0 No excessive deferral. 1 Indicates that the current frame has been deferred for an excessive period of time. Applicable only in half duplex mode. The value of this period in bit times is calculated in the following ways: For 10/100 Mbps operation it is: 2 x (maxFrameSize x 8) bit times. For 1000 Mbps operation it is: 2 x (burstlimit+maxFrameSize x 8+headerSize) bit times.	R
10	Excessive collisions	0 Less than 16 collisions. 1 Indicates that the current frame transmission had ended with a collision on the 16th consecutive attempt. Applicable only in half-duplex mode.	R
11	Late collision	0 No late collision. 1 Frame collided outside of the collision window. Applicable only in half-duplex mode.	R
12	Multiple collision	0 More than 1 but less than 16 collisions did not occur. 1 Transmitted frame collided more than once but less than 16 times. Applicable only in half-duplex mode.	R
13	Single collision	0 Single collision did not occur. 1 Activates if transmitted frame collided once. Applicable only in half-duplex mode.	R

Bits	Bit Name	Bit Description	Mode
14	Underrun	0 Underrun did not occur. 1 Frame transmission was aborted because of underrun; data from the Transmit FIFO was not valid in time to allow continuous data transmission on the MII/GMII interface.	R
15	SQE	0 Signal Quality Error did not occur. 1 Signal Quality Error test failed during packet transmission. Applicable only in half -duplex mode during 10 Mbps operation.	R

27.3.1.2 Early Packet Termination during Transmit

EMAC can initiate early packet termination during transmit, terminating packet transmission before MAL finishes transferring all packet data from memory to the EMAC transmit FIFO. Early packet termination is typically used when error conditions force the EMAC to abort a transmission.

EMAC performs early termination on the MAL interface if any of the following conditions occur:

- Underrun in the transmit FIFO.
- Excessive collisions.
- Excessive deferral.
- Late collision.

27.3.1.3 Empty Packets

EMAC treats empty packets as if a normal packet had been written, but does not write data to the transmit FIFO. A status word of all 0s is returned after an empty packet. EMAC expects that for quadword-aligned packets, MAL activates the related word transfer indication during the last data transfer, rather than providing an empty packet indication.

27.3.1.4 Automatic Retransmission of Colliding Packets

EMAC automatically retransmits packets that collide on the MII/GMII interface. The transmit FIFO always preserves the first 64 (in 10/100 Mbps) or 512 (in Gigabit Ethernet media) bytes of a packet until it receives an indication that the collision window has passed. Otherwise, if a collision was detected within the collision window, the packet is retransmitted without a new request from MAL.

27.3.1.5 Inter-Packet Gap (IPG) Tuning

EMAC supports user-programmable IPG length using the EMACx_IPGVR register, which contains one-third of the IPG value. Changing the contents of EMACx_IPGVR enables the user to adjust the fairness or aggressiveness of EMAC on the medium. Programming a lower number (but not less than four) causes EMAC becomes more aggressive on the media. This can result in EMAC capturing the network by forcing less aggressive nodes to defer. Programming a larger number of bit times causes EMAC to becomes less aggressive on the network; it might defer more often than normal. EMAC performance might decrease as the IPG period is increased from the default value, but the resulting behavior can improve media performance by reducing the occurrence of collisions.

27.3.1.6 Full-Duplex Operation

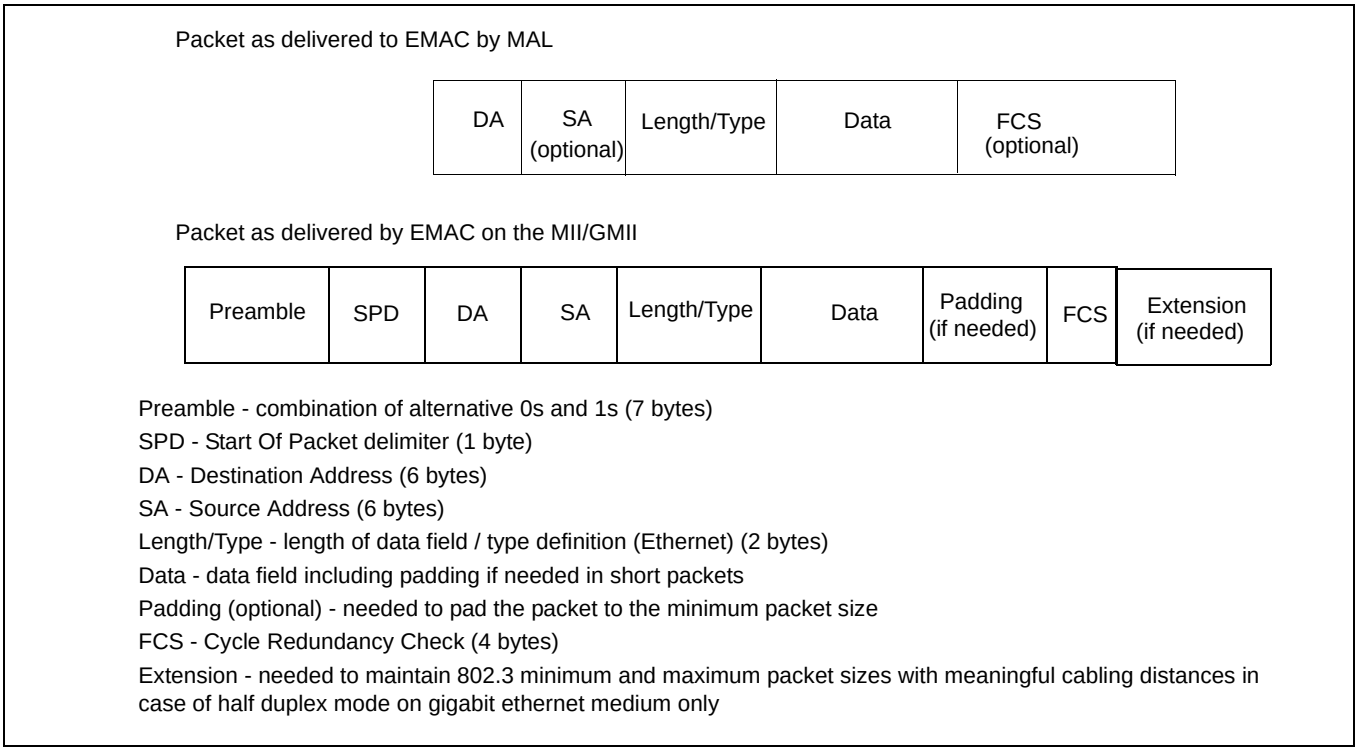
Full-duplex operation allows simultaneous transmit and receive activity on the MII/GMII interface. Software can set EMACx_MR1[FDE] = 1 to enable full-duplex mode. During full-duplex operation, the following changes occur in EMAC functionality.

- Transmission is not deferred while receive is active.
- The IPG counter, which controls transmit deferral during the IPG between back-to-back transmits, is started when transmit activity for the first packet ends, instead of when transmit and carrier activity end.
- SQE test is not performed.
- Collision indication is ignored.

27.3.1.7 Packet Content Configuration Options

EMAC can modify the content of the packet coming from MAL before issuing it to the MII/GMII interface. *Figure 27-8* illustrates possible changes in the transmit packet format.

Figure 27-8. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)


Packet as delivered to EMAC by MAL
DA
SA (optional)
Length/Type
Data
FCS (optional)
Packet as delivered by EMAC on the MII/GMII
Preamble
SPD
DA
SA
Length/Type
Data
Padding (if needed)
FCS
Extension (if needed)
Preamble - combination of alternative 0s and 1s (7 bytes)
SPD - Start Of Packet delimiter (1 byte)
DA - Destination Address (6 bytes)
SA - Source Address (6 bytes)
Length/Type - length of data field / type definition (Ethernet) (2 bytes)
Data - data field including padding if needed in short packets
Padding (optional) - needed to pad the packet to the minimum packet size
FCS - Cycle Redundancy Check (4 bytes)
Extension - needed to maintain 802.3 minimum and maximum packet sizes with meaningful cabling distances in case of half duplex mode on gigabit ethernet medium only

The following options, unless mutually exclusive, can be set for each packet, and can be provided as a part of command write transactions from MAL (see *MAL TX Descriptor Control/Status Field* on page 703).

- Automatic data padding for short transmit packets

  EMAC pads the transmit packet with extra bytes between the data and the FCS field to reach a total length of 64 bytes (including FCS). This feature is supported only when the packet coming from the transmit FIFO does not contain the FCS. Automatic padding enables software to avoid sending padding as a part of the packet data field, and, therefore, reduces the amount of data transferred on the system bus during the short packet transmission.

- Source address insertion

  EMAC adds the source address (SA) field to the transmitted packet. EMAC uses the contents of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) registers for the source address value. *This option is mutually exclusive with source address replacement.*

- Source address replacement

  EMAC replaces the source address received from the transmit FIFO with the contents of EMACx_LSAH and EMACx_LSAL. *This option is mutually exclusive with source address insertion*.

- Add four FCS bytes

  EMAC calculates the FCS for the transmitted packet. The FCS is appended to data coming from the Transmit FIFO or padding bytes (if added).

- VLAN Tag insertion

  EMAC adds the content of the VLAN Tag field to the transmitted packet (see *VLAN Support* on page 715). EMAC uses the content of the VLAN TPID (EMACx_VTPID) and VLAN TCI (EMACx_VTCI) registers for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag replacement*.

- VLAN Tag replacement.

  EMAC replaces the content of the VLAN Tag field in the transmitted packet. EMAC uses the contents of EMACx_VTPID and EMACx_VTCI for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag insertion*.

Table 27-2 summarizes the possible options for adding FCS and SA to the transmitted packet.

Table 27-2. FCS/SA Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	**Insert SA**	**Replace SA**	**Add FCS**	**Add SA**	**Replace SA**
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

Table 27-3 summarizes the possible options for adding FCS and padding to the transmitted packet.

Table 27-3. FCS/Pad Enable - Possible Configurations

Configuration Options		EMAC Action	
Generate FCS	**Generate Pad**	**Add FCS**	**Add Pad**
0	Don't care	N	N
1	0	Y	N
1	1	Y	Y

Table 27-4 summarizes the possible options for adding FCS and VLAN Tag to the transmitted packet.

Table 27-4. FCS/VLAN Tag Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	**Insert VLAN Tag**	**Replace VLAN Tag**	**Add FCS**	**Insert VLAN Tag**	**Replace VLAN Tag**
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

27.4 EMAC Receive Operation

The receive part of EMAC is responsible for receiving packets coming from the physical layer (PHY) device (via the MII/GMII) and forwarding them to the receive channel of the attached MAL. At the end of the reception process, EMAC provides a status/error word that enables software to monitor the receive operation.

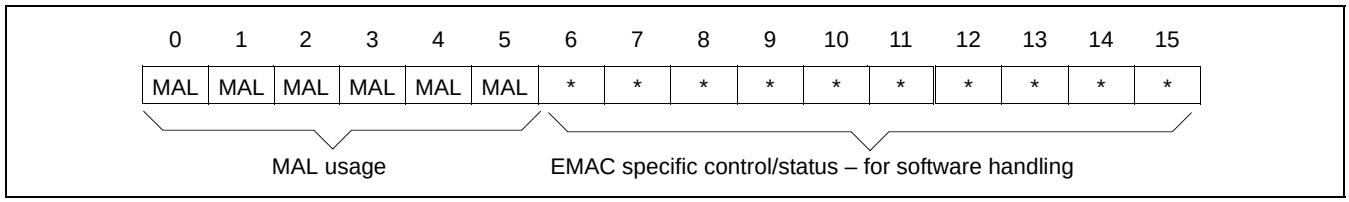
27.4.1 EMAC – MAL RX Packet Transfer Flow

EMAC initiates request for service from MAL when the number of occupied entries in the receive FIFO reaches the low water mark specified in EMACx_RWMR[RLWM].

27.4.2 MAL RX Descriptor Status

For each packet that is received, MAL obtains status from EMAC after reception is complete, and writes this information into the buffer descriptor status/control field. Software uses this information to monitor the status of received packets. See *Buffer Descriptor* on page 652 for more information on the buffer descriptor structure.

Figure 27-9. MAL RX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See *RX Status/Control Field Format* on page 661.	R
RX Status Information (Read Access)			
6	Overrun Error	0 No overrun error. 1 EMAC detected an overrun error. An overrun error occurs if the flow of received data to the RX FIFO is corrupted because of insufficient empty space.	R
7	Pause Packet	0 Received packet is not a control pause packet. 1 Received packet is a control pause packet.	R
8	Bad Packet	0 No packet errors. 1 Early termination caused by packet error.	R
9	Runt Packet	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal greater than ShortEventMax Time constant and less than collision windows.	R
10	Short Event	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal was less than ShortEventMaxTime constant	R
11	Alignment Error	0 Received packet length OK. 1 Received packet length not an integral number of octets.	R
12	Bad FCS	0 FCS OK. 1 The FCS value does not match the FCS value calculated by EMAC.	R

Bits	Bit Name	Bit Description	Mode
13	Packet Too Long	0 Received packet length OK. 1 Received packet length exceeds maximum packet length. 1518 octets for standard packet (is checked only when the length/type field of the transmitted packet contained length value and Jumbo support is disabled) 1522 octets for VLAN tagged packet (is checked only when the length/type field of the transmitted packet contained length value and Jumbo support is disabled) 9018 octets for jumbo packet (with no VLAN support) 9022 octets for VLAN tagged Jumbo packet **Note:** The data following the maximum packet length is not transferred to MAL	R
14	Out of Range Error	0 Received packet length field value OK. 1 Received packet length field value greater than maximum allowed LLC data size. The maximum allowed logical link control (LLC) data size is greater than 1500 and less than 1536.	R
15	In Range Error	Refer to *Table 27-5* for a description of conditions for activating this status bit.	R

Table 27-5. In Range Length Error Behavior for Various Packet Lengths

Programmed Length (Bytes)	Actual Length	EMAC Action
Less than 46 (42 if VLAN Tagged packet)	Differs from 46 (42 in case of VLAN Tagged packet)	In range length error is activated
Less than 46 (42 if VLAN Tagged packet)	46 (42 in case of VLAN Tagged packet)	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Equals the length field value	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Differs from the length field value	In range length error is activated

27.4.3 Early Packet Termination during Receive

Early packet termination occurs when packet reception is aborted by EMAC before the packet data transfer to MAL is completed.

EMAC performs early termination in the following cases.

- An overrun occurs in the receive FIFO
- Packet is too long and the Receive Oversize Packet option is not enabled (EMACx_RMR[ROP] = 0)

27.4.4 Discarding Packets During Receive

Received packets can be discarded if certain error conditions are detected. EMAC behavior depends on whether the packet to be discarded is already being output to MAL. If the packet containing the error is not being provided to MAL when the discard condition is detected, the packet is flushed from the Receive FIFO. In this case, EMAC does not provide status information to MAL. If the packet containing the error is already being output to MAL, EMAC initiates an early packet termination procedure, as described in *Early Packet Termination during Receive* on page 710

Each receive discard condition can be individually controlled using appropriate settings, as described in *Receive Mode Register (EMACx_RMR)* on page 726.

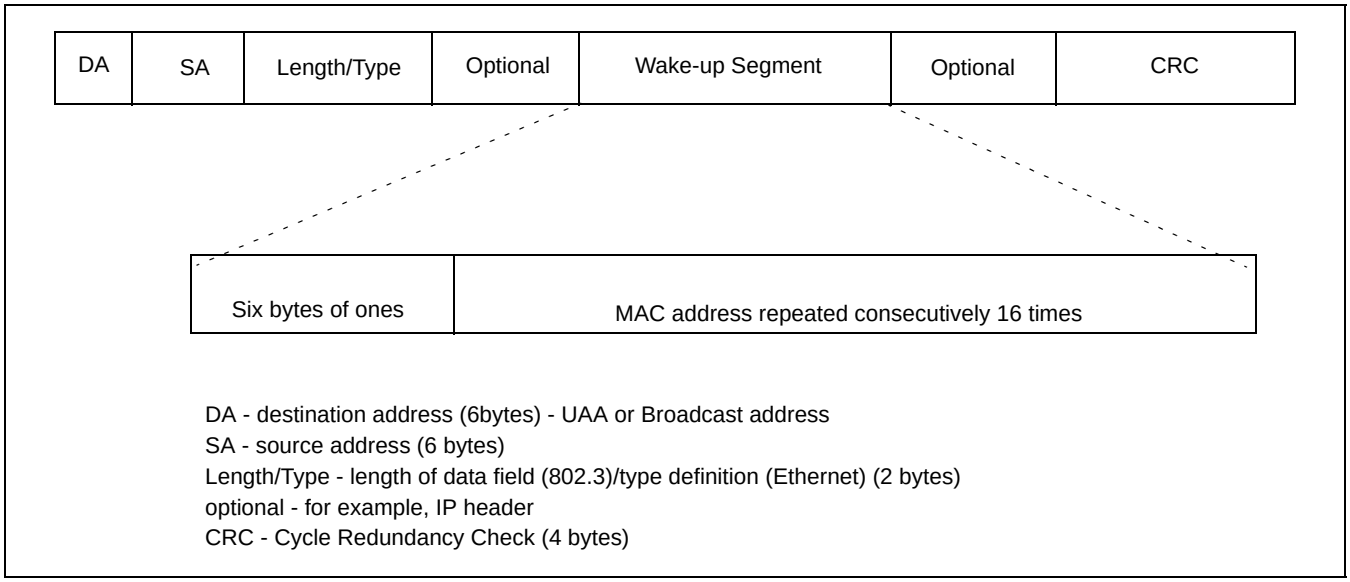
27.4.5 Wakeup On Lan (WOL) Support

WOL logic in EMAC supports WOL technology, an industry standard in the Wired for Management (WFM) Specification. WOL remotely awakens sleeping or powered-off nodes on a network. To wake up a node, a specific packet, called a *magic packet*, is sent to the network node. When so configured, EMAC monitors the incoming bit stream for magic packets.

The magic packet, also called a wake-up packet, contains a unique data field not normally expected in LAN traffic. When a WOL-enabled adapter on a powered-off client decodes this data field, a wake-up signal is generated. In the PPC440EPx/GRx, with WOL mode enabled, EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1).

Figure 27-10 shows the wake-up packet format. The key to the wake-up packet is the MAC address of the target client, which is repeated 16 times. This pattern of 16 addresses in the data field is not expected to occur in any packet except the wake-up packet.

Figure 27-10. Wake-Up Packet Format


DA
SA
Length/Type
Optional
Wake-up Segment
Optional
CRC
Six bytes of ones
MAC address repeated consecutively 16 times
DA - destination address (6bytes) - UAA or Broadcast address
SA - source address (6 bytes)
Length/Type - length of data field (802.3)/type definition (Ethernet) (2 bytes)
optional - for example, IP header
CRC - Cycle Redundancy Check (4 bytes)

The destination address can be a specific address, called the universally administered address (UAA), or a broadcast address. If the destination address is a UAA, the wake-up packet is sent only to the client at that address. However, because the client is powered off and is no longer transmitting, some protocols remove the client MAC address from routing tables and internal caches at other nodes. In this case, wake-up packets addressed to a target client are discarded because nodes and routers do not know where to send them. The solution to this problem is to use a broadcast address. A directed broadcast has a valid network address and a broadcast host address. Network routers and nodes forward directed broadcasts to the appropriate network, where it is seen as a MAC-level broadcast and detected by the powered-off client.

27.4.5.1 EMAC WOL Support

EMAC enters WOL mode when EMACx_MR0[WKE] = 1.

WOL mode should only be changed while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0. After EMACx_MR0[WKE] = 1, EMACx_MR0[RXE] can be set to 1.

A reset (soft or hard) should be issued before programming EMAC to WOL mode. In WOL mode, EMAC does not propagate any received packets to MAL. Also, EMAC transmit channels do not request data from MAL.

27.5 Flow Control

For efficient system performance, each EMAC implements full-duplex flow control by handling specific MAC control packets contained in the pause opcode. EMAC supports flow control as defined in the IEEE 802.3x-1997 standard.

27.5.1 MAC Control Packet

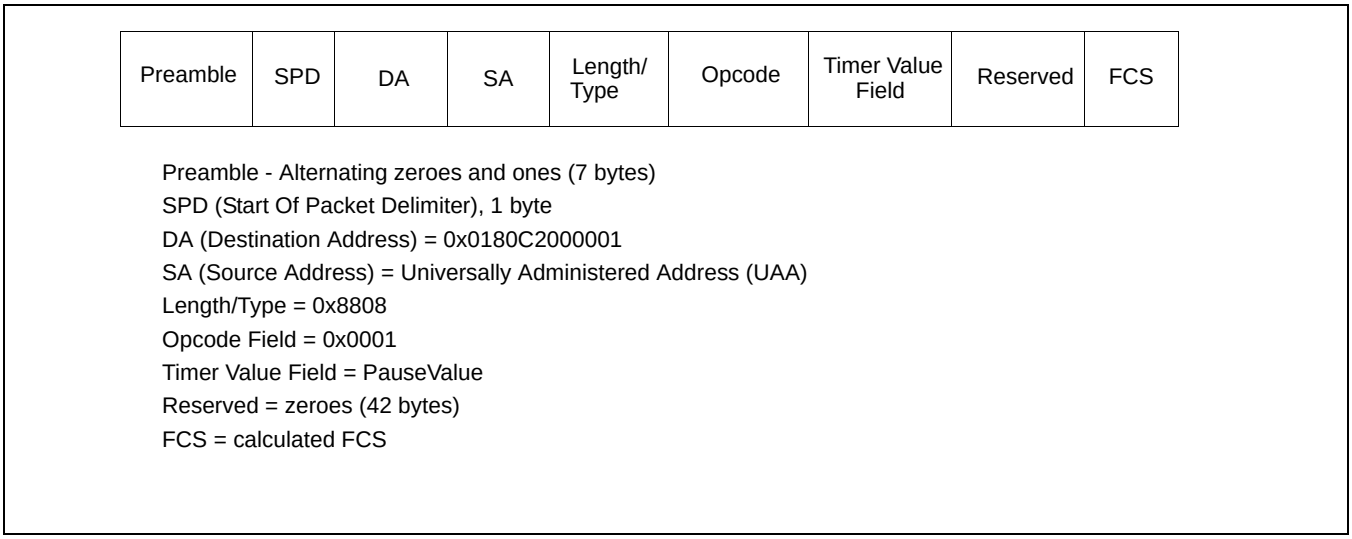
The flow control mechanism enables receive FIFO control logic to automatically notify the node transmitting packets to suspend transmission for a defined period of time. The pause control packet has a fixed length defined as follows:

MinFrameSize – 32 bits (60 bytes)

MAC control packets have a unique value of 0x8808 in the length/type field, and share the same packet format as normal Ethernet packets, except that the data field consists of an opcode field and a parameter field. The opcode field contains an opcode command and the parameter field contains a value associated with the opcode command (0x0001). The only opcode command defined by IEEE 802.3x is the pause opcode; the parameter field for the pause opcode defines the pause time. MAC control packets containing a pause opcode, also called pause packets, should have a destination address equal to a reserved multicast address. The reserved multicast address is 0x0180C2000001.

Note: This reserved multicast address should also be mapped into one of 64 categories corresponding to 64 bits stored in the EMACx_GAHT1EMACx_GAHT4 registers.

Figure 27-11. Control Packet Format



The timer value field contains the value of the delay interval in resolution of *pause_quanta*, defined in IEEE 802.3x as follows: "MAC Control Parameter[s] (pause_time) is a 2-octet, unsigned integer containing the length of time for which the receiving station is requested to inhibit data packet transmission. The field is transmitted most-significant octet first, and least-significant octet second. The pause_time is measured in units of pause_quanta, equal to 512 bit times. The range of possible pause_time is 0 to 65535 pause_quanta."

27.5.2 Control Packet Transmission

Two options initiate the pause packet transmission from EMAC. The transmitted pause packet forces the node, with the destination address specified, to temporarily suspend the transmission of packets to EMAC.

- Software initiated

  The packet transferred to EMAC by MAL for transmission is a pause packet created by software. EMAC transmits this as a normal packet.

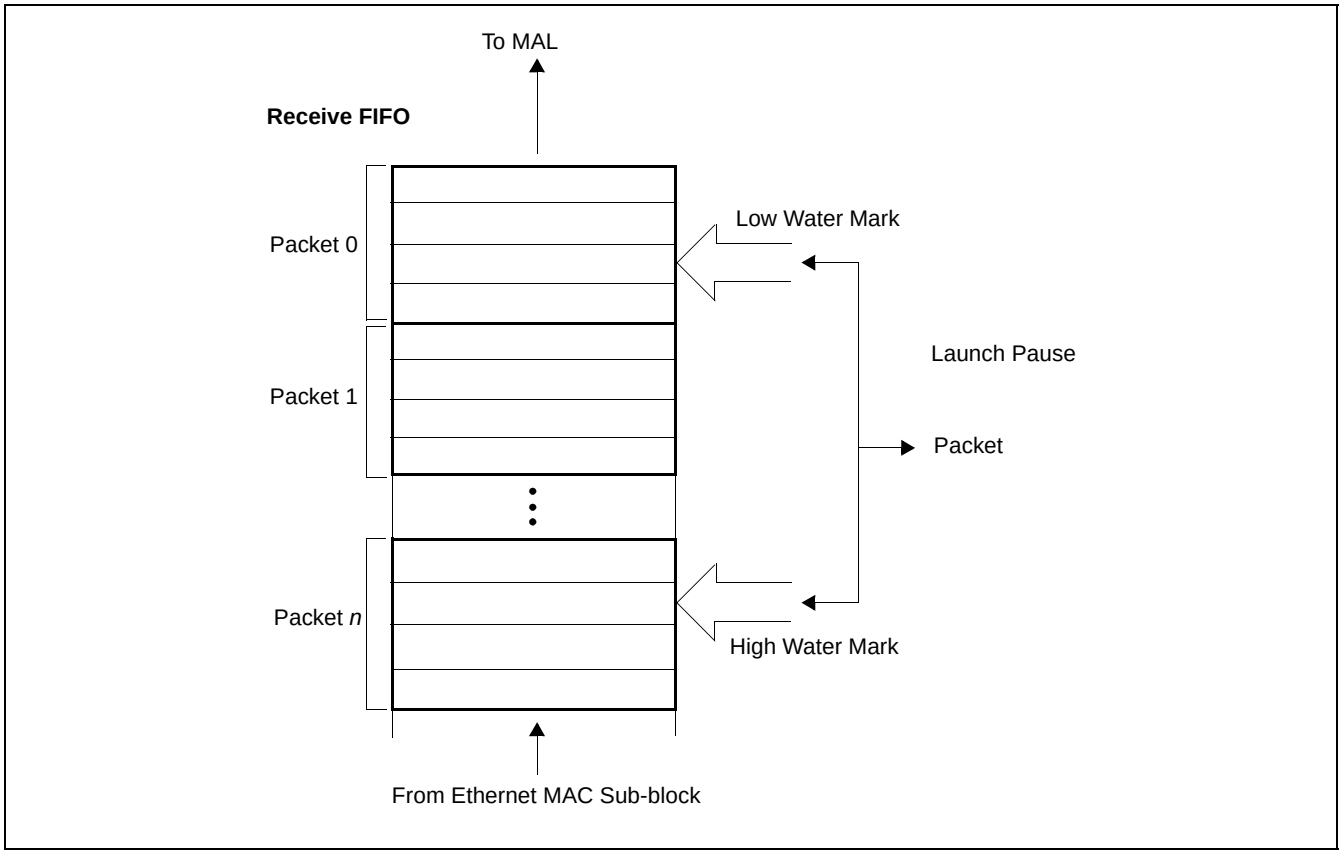
- Automatic flow control initiated

  The EMAC integrated flow control mechanism detects the need for and then transmits a control (pause) packet automatically. When building the control packet, EMAC obtains the SA (source address) field from EMACx_IAHR and EMACx_IALR, and the timer value from the Pause Timer Register (EMACx_PTR[TVF]). The contents of the other fields in the packet are shown in *Figure 27-11*.

27.5.3 Integrated Flow Control

To enable integrated flow control in full-duplex mode, set EMACx_MR1[EIFC] = 1. When the receive FIFO reaches a predefined threshold level (called a high water mark and specified by EMACx_RWMR[RHWM]), an internal request for control (pause) packet transmission is activated. EMAC sends a control (pause) packet when a new packet enters the receive FIFO and the number of vacant entries in the Receive FIFO is less than the high water mark. When the receive FIFO reaches another predefined threshold level (the low water mark, specified by EMACx_RWMR[RLWM]), a new internal request for a pause packet transmission, with a pause timer value of 0, is activated. EMAC sends a pause packet, with a pause timer value of 0, only once, and only if a pause packet with a non-zero value in the pause timer was transmitted earlier.

Figure 27-12. Integrated Flow Control Mechanism

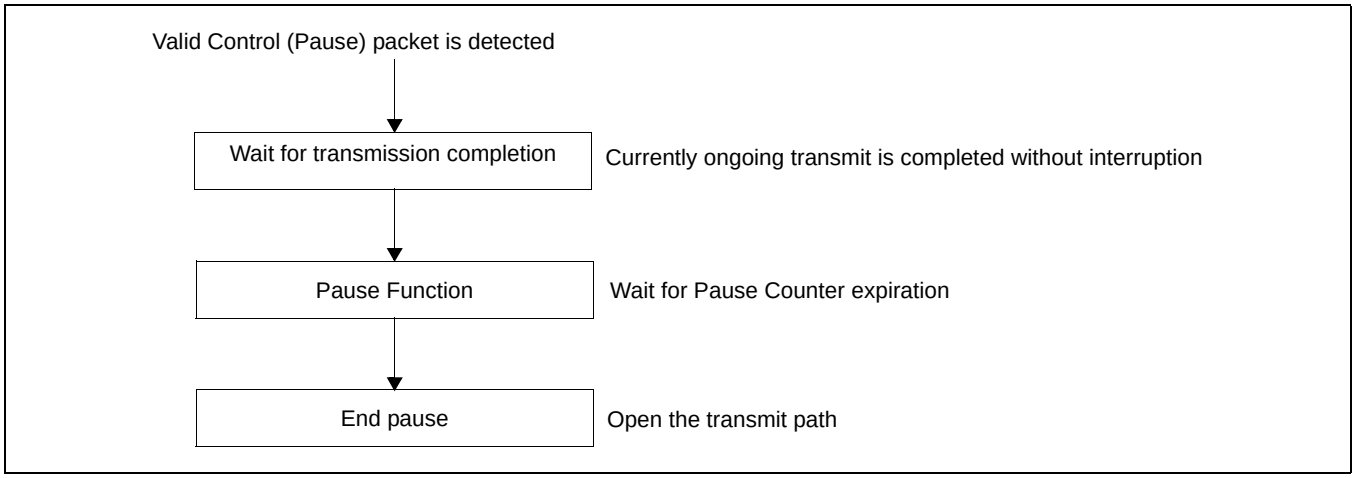


27.5.4 Control Packet Reception

In the receive path, the EMAC can be configured to respond to pause packets, or ignore them, as specified by EMACx_MR1[APP]. In order for the EMAC to process incoming Pause Packets, EMACx_RMR[MAE] and EMACx_RMR[BAE] bits must also be set to 1. When response to pause packets is enabled and EMAC detects a valid MAC control packet with a pause opcode, the EMAC stores the value of the timer value field. The received packet is considered a valid control packet only if no error was detected during the packet reception. If, at the end of packet reception, the packet is considered valid, EMAC launches a pause operation state machine, as specified in the IEEE P802.3x standard. *Figure 27-13* illustrates the pause operation state machine.

If a control (pause) packet is received while another packet is transmitted, the ongoing transmission process is completed and the transmitter is paused. If other packets are in the transmit FIFO, their transmission is delayed until the pause timer expires. EMAC normally does not pass the MAC control packets to MAL unless EMACx_RMR[PPP] = 1.

Figure 27-13. Pause Operation State Machine


Valid Control (Pause) packet is detected
Wait for transmission completion
Currently ongoing transmit is completed without interruption
Pause Function
Wait for Pause Counter expiration
End pause
Open the transmit path

In the Pause Function state, EMAC decrements its internal pause timer, which was set to the timer value field of the received control packet.

Note 1: The transmission of control (pause) packets is not affected by the reception of a receive control (pause) packet. Received control (pause) packets inhibit only the transmission of regular packets from the Transmit FIFO.

Note 2: Receipt of a new valid control (pause) packet causes the pause timer of EMAC to be reloaded with the contents of the timer value field of the recently received packet, regardless of the current pause timer setting. This indicates new pause operations take precedence over earlier pause operations.

27.6 VLAN Support

EMAC can handle VLAN tagged packets, as specified in IEEE Draft P802.3ac/D1.0a standard when EMACx_MR1[VLE] = 1.

A VLAN tagged frame is an extension of the standard MAC packet. The extension for VLAN tag support consists of a 4-octet VLAN tag inserted between the end of the source address and the beginning of the length/type fields of the MAC packet.

The VLAN tag consists of two fields:

- A 2-octet constant Type field value equal to the VLAN Tag Protocol Identifier (0x8100)
- A 2-octet field containing Tag Control Information (TCI)

The MAC client data and FCS fields of the basic MAC packet follow the VLAN tag. The length of the packet is extended by four octets by the VLAN tag (up to 1522 bytes and 9022 bytes for standard and jumbo frame respectively). The FCS is calculated over all fields from the destination address through the end of the MAC client data or pad (if present); that is, all fields except the preamble, SPD, and FCS.

Figure 27-14. VLAN Tagged Packet Format

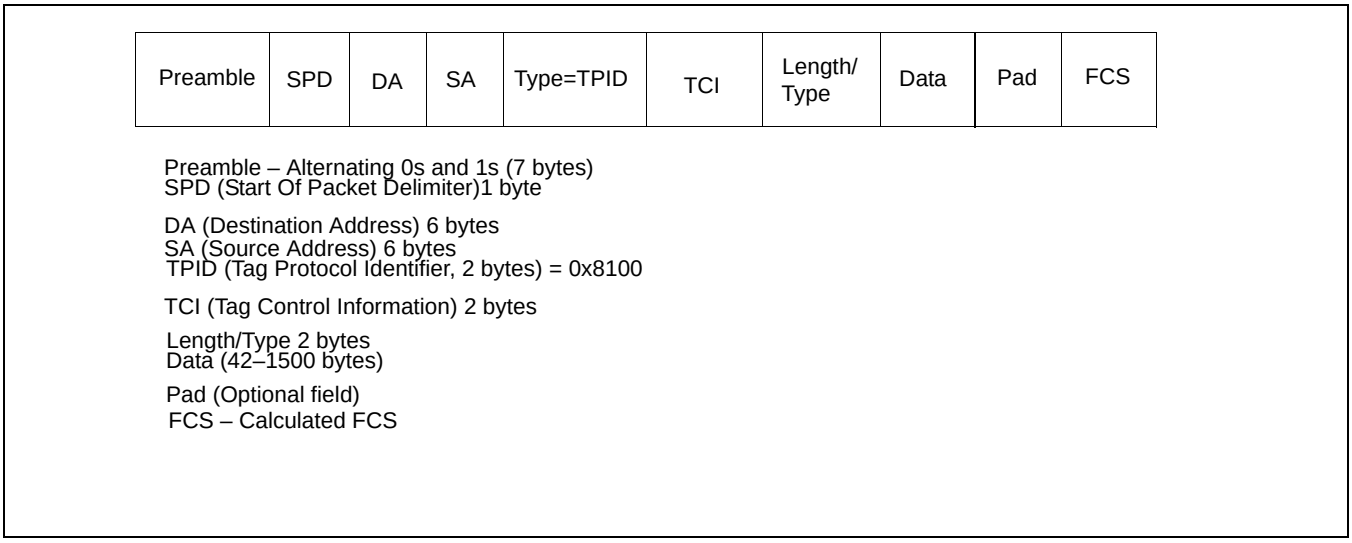
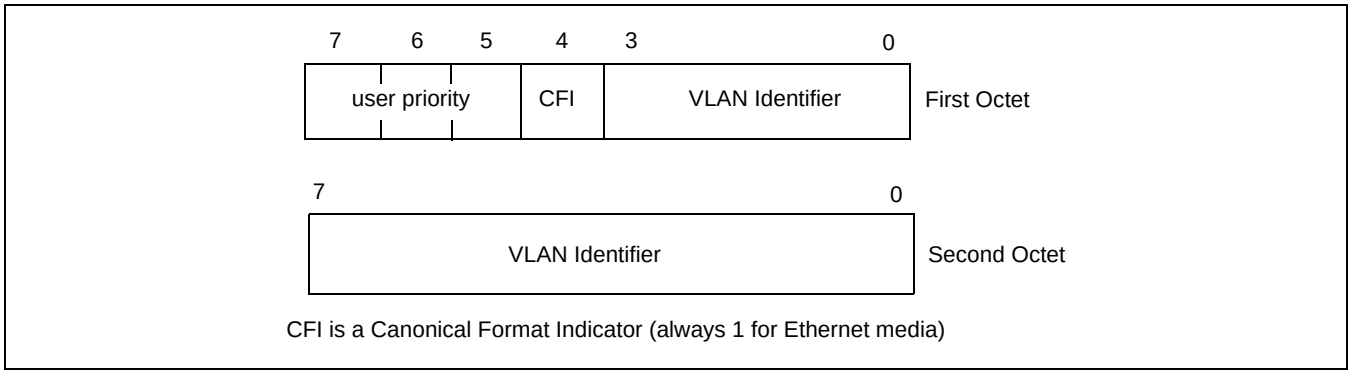


Figure 27-15. Tag Control Information Field Structure



27.6.1 VLAN Tagged Packet Transmission

When EMACx_MR1[VLE] = 1, the following configuration options are available, depending on the content of the appropriate bits in the MAL control word (See *MAL TX Descriptor Control/Status Field* on page 703.)

- The Generate FCS bit (bit 6) is not set or both Insert VLAN Tag and Replace VLAN Tag bits (bits 10 and 11, respectively) are not set: EMAC transmits the packet without any changes
- Bit 6 is set and bit 10 is also set: EMAC will insert TPID and TCI for the transmitting packet using the content of EMACx_VTPID and EMACx_VTCI, respectively
- Bit 6 is set and bit 11 is also set: EMAC will replace TPID and TCI for the transmitting packet using the content of EMACx_VTPID and EMACx_VTCI, respectively

27.6.2 VLAN Tagged Packet Reception

If EMACx_MR1[VLE] = 1, the EMAC parses the VLAN Tag unique type/length in the incoming packet during the receive process. If the VLAN Tag is equal to the value stored in EMACx_VTPID, EMAC continues the receive process and allows the received packet to contain up to 1522 octets and 9022 when Jumbo packets support is enabled. Otherwise, the receive process is continued unless the length is greater than 1518 bytes 9018 when Jumbo packets support is enabled.

27.6.3 Address Match Mechanism

The address match (or filtering) mechanism is a hardware aid that reduces the average amount of CPU cycles required to determine whether an incoming packet should be accepted.

EMAC uses various address filters for incoming packets by using the following address recognition modes.

- Individual mode (also referred to as physical)
- Multicast mode (also referred to as group)
- Broadcast mode (an all-ones group address)
- Promiscuous mode
- Promiscuous multicast mode
- WOL mode

A flowchart for address recognition of received packets is shown in *Figure 27-16* on page 719. If the least significant bit (LSb) of the first byte of the destination address (DA) is 0, the packet is considered individual. If the first bit received is 1, the packet is considered multicast. When the DA field contains all 1s, the packet is broadcast, a special type of multicast.

27.6.3.1 Non-WOL Mode

When EMAC operates in single individual mode (EMACx_RMR[IAE] = 1), the DA of the received packet is compared to the physical address stored in EMACx_IAHR and EMACx_IALR.

When EMAC operates in multiple individual mode (EMACx_RMR[MIAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) to determine whether or not to accept the packet.

When EMAC operates in promiscuous mode (EMACx_RMR[PME] = 1), all properly formed packets are received, regardless of the content of the DA field.

When EMAC operates in multicast promiscuous mode (EMACx_RMR[PMME] = 1), all multicast packets are received, regardless of the content of the DA field.

When EMAC operates in broadcast address mode (EMACx_RMR[BAE] = 1), EMAC performs an address compare on received packets with broadcast addresses.

When EMAC operates in multicast address mode (EMACx_RMR[MAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) as in multiple individual mode, in order to determine whether or not the packet should be accepted.

The logical address filter hardware implements a hash code searching technique commonly used by programmers. The hardware maps the DA of the incoming packet into one of 64 categories corresponding to 64 bits stored in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers. The hardware accepts or rejects the packet, depending on the state of the corresponding bit in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers corresponding to the selected category.

Figure 27-17 on page 720 shows the details of the hardware mapping algorithm. The example depicts multiple individual address mode, but with changes can be used for the multicast address mode.

If the most significant bit (MSb) of an incoming address is 0, the address is individual and is passed to the individual address filter. If the MSb of an incoming address is 1, the address is multicast and is passed to the multicast address filter. The individual/multicast address filter is a 64-bit mask composed of the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers (each register contains 16 bits of a 64-bit mask). The incoming address is sent through the FCS circuit. After the 48 address bits have gone through the FCS circuit, the high-order 6 bits of the resulting FCS (32-bit CRC) are used to select a the 64-bit positions in the individual/multicast address filter. If the selected filter bit is 1, the address is accepted.

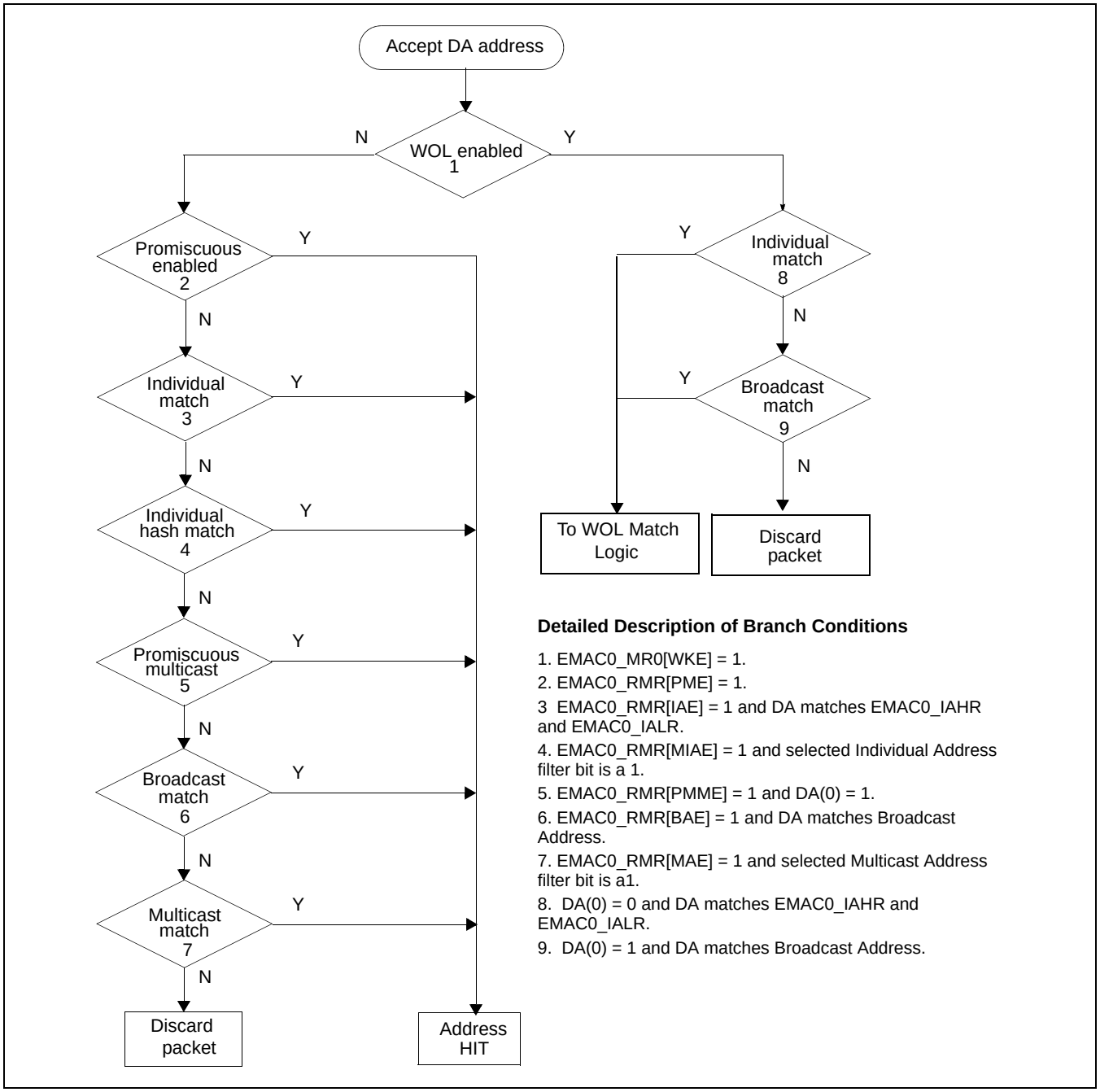
Note: The individual/multicast address filter ensures only that there is a possibility that the incoming packet belongs to this node. To determine if the packet belongs to the node, the incoming individual/multicast address propagated to the main memory is compared by software to the list of logical addresses to be accepted by this node.

For software, the task of mapping an individual/multicast address to one of 64 bit positions requires a program that uses the same CRC algorithm to calculate the hash.

27.6.3.2 WOL Mode

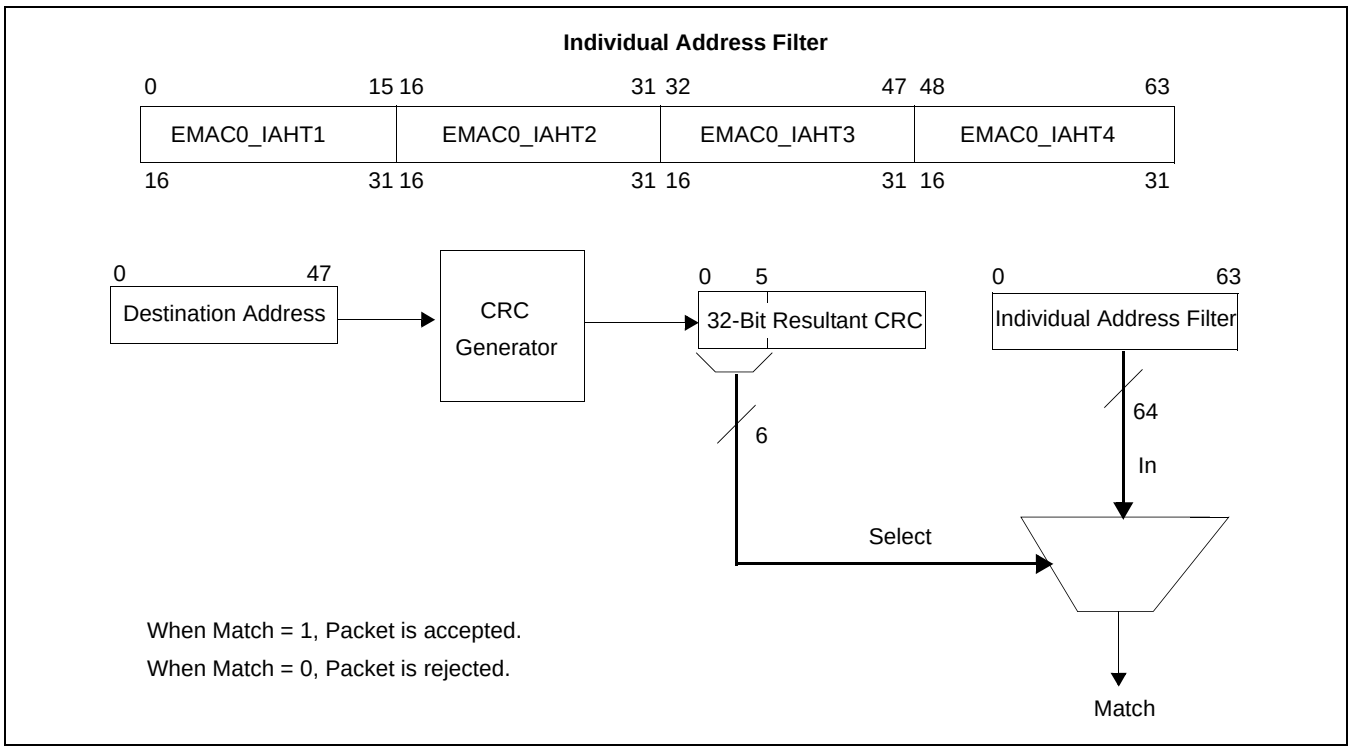
In WOL mode (EMACx_MR0[WKE] = 1), EMAC operates only with the broadcast or individual address in the destination address field.

Figure 27-16. Receive Address Recognition Flowchart



The example in *Figure 27-17* shows that the received individual address maps into category 24, and that bit 8 in EMACx_IAHT2 is set. The match indication is activated and the packet should be accepted.

Figure 27-17. Ethernet Address Filter Operation



27.7 EMAC Registers

This section describes the EMAC registers, which are listed in *Table 27-6* for EMAC0 and *Table 27-7* for EMAC1. In the register descriptions, the registers are prefixed "EMACx" to denote the identical register implementations in each EMAC. The EMAC registers are accessed using the OPB slave interface. Access to these memory-mapped I/O (MMIO) registers should be word-aligned.

Note: It is required to perform a soft reset of EMAC before initialization as shown in EMACx Mode Register 0 (EMACx_MR0[SRST]).

Table 27-6. EMAC0 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC0_MR0	0x1 EF600E00	See description in *Scenario 1* on page 740	0xC0000000	R/W	725
EMAC0_MR1	0x1 EF600E04	Reset	0x00000000	R/W	723
EMAC0_TMR0	0x1 EF600E08	See description on page 725	0x00000007	R/W	725
EMAC0_TMR1	0x1 EF600E0C	See description on page 726	0x7807C000	R/W	726
Note: See *Reset and Initialization* on page 740 for definitions of letters in the Write Access column.					

Table 27-6. EMAC0 Register Summary (continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC0_RMR	0x1 EF600E10	Reset	0x00000007	R/W	726
EMAC0_ISR	0x1 EF600E14	Always	0x00000000	R/W	728
EMAC0_ISER	0x1 EF600E18	Reset	0x00000000	R/W	730
EMAC0_IAHR	0x1 EF600E1C	Reset, R, T	0x00000000	R/W	732
EMAC0_IALR	0x1 EF600E20	Reset, R, T	0x00000000	R/W	732
EMAC0_VTPID	0x1 EF600E24	Reset, R, T	0x00008808	R/W	733
EMAC0_VTCI	0x1 EF600E28	Reset, R, T	0x00000000	R/W	733
EMAC0_PTR	0x1 EF600E2C	Reset, T	0x0000FFFF	R/W	733
EMAC0_IAHT1	0x1 EF600E30	Reset, R	0x00000000	R/W	734
EMAC0_IAHT2	0x1 EF600E34	Reset, R	0x00000000	R/W	734
EMAC0_IAHT3	0x1 EF600E38	Reset, R	0x00000000	R/W	734
EMAC0_IAHT4	0x1 EF600E3C	Reset, R	0x00000000	R/W	734
EMAC0_GAHT1	0x1 EF600E40	Reset, R	0x00000000	R/W	734
EMAC0_GAHT2	0x1 EF600E44	Reset, R	0x00000000	R/W	734
EMAC0_GAHT3	0x1 EF600E48	Reset, R	0x00000000	R/W	734
EMAC0_GAHT4	0x1 EF600E4C	Reset, R	0x00000000	R/W	734
EMAC0_LSAH	0x1 EF600E50	Not applicable	0x00000000	R	734
EMAC0_LSAL	0x1 EF600E54	Not applicable	0x00000000	R	734
EMAC0_IPGVR	0x1 EF600E58	Reset, T	0x00000004	R/W	735
EMAC0_STACR	0x1 EF600E5C	See description on page 735	0x00001000	R/W	735
EMAC0_TRTR	0x1 EF600E60	See description on page 736	0x00000000	R/W	736
EMAC0_RWMR	0x1 EF600E64	Reset	0x04000800	R/W	737
EMAC0_OCTX	0x1 EF600E68	Not applicable	0x00000000	R	738
EMAC0_OCRX	0x1 EF600E6C	Not applicable	0x00000000	R	738
Note: See *Reset and Initialization* on page 740 for definitions of letters in the Write Access column.					

Table 27-7. EMAC1 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC1_MR0	0x1 EF600F00	See description in *Scenario 1* on page 740	0xC0000000	R/W	725
EMAC1_MR1	0x1 EF600F04	Reset	0x00000000	R/W	723
EMAC1_TMR0	0x1 EF600F08	See description on page 725	0x00000007	R/W	725
Note: See *Reset and Initialization* on page 740 for definitions of letters in the Write Access column.					

Table 27-7. EMAC1 Register Summary (continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC1_TMR1	0x1 EF600F0C	See description on page 726	0x7807C000	R/W	726
EMAC1_RMR	0x1 EF600F10	Reset	0x00000007	R/W	726
EMAC1_ISR	0x1 EF600F14	Always	0x00000000	R/W	728
EMAC1_ISER	0x1 EF600F18	Reset	0x00000000	R/W	730
EMAC1_IAHR	0x1 EF600F1C	Reset, R, T	0x00000000	R/W	732
EMAC1_IALR	0x1 EF600F20	Reset, R, T	0x00000000	R/W	732
EMAC1_VTPID	0x1 EF600F24	Reset, R, T	0x00008808	R/W	733
EMAC1_VTCI	0x1 EF600F28	Reset, R, T	0x00000000	R/W	733
EMAC1_PTR	0x1 EF600F2C	Reset, T	0x0000FFFF	R/W	733
EMAC1_IAHT1	0x1 EF600F30	Reset, R	0x00000000	R/W	734
EMAC1_IAHT2	0x1 EF600F34	Reset, R	0x00000000	R/W	734
EMAC1_IAHT3	0x1 EF600F38	Reset, R	0x00000000	R/W	734
EMAC1_IAHT4	0x1 EF600F3C	Reset, R	0x00000000	R/W	734
EMAC1_GAHT1	0x1 EF600F40	Reset, R	0x00000000	R/W	734
EMAC1_GAHT2	0x1 EF600F44	Reset, R	0x00000000	R/W	734
EMAC1_GAHT3	0x1 EF600F48	Reset, R	0x00000000	R/W	734
EMAC1_GAHT4	0x1 EF600F4C	Reset, R	0x00000000	R/W	734
EMAC1_LSAH	0x1 EF600F50	Not applicable	0x00000000	R	734
EMAC1_LSAL	0x1 EF600F54	Not applicable	0x00000000	R	734
EMAC1_IPGVR	0x1 EF600F58	Reset, T	0x00000004	R/W	735
EMAC1_STACR	0x1 EF600F5C	See description on page 735	0x00001000	R/W	735
EMAC1_TRTR	0x1 EF600F60	See description on page 736	0x00000000	R/W	736
EMAC1_RWMR	0x1 EF600F64	Reset	0x04008000	R/W	737
EMAC1_OCTX	0x1 EF600F68	Not applicable	0x00000000	R	738
EMAC1_OCRX	0x1 EF600F6C	Not applicable	0x00000000	R	738

Note: See *Reset and Initialization* on page 740 for definitions of letters in the Write Access column.

27.7.1 Mode Register 0 (EMACx_MR0)

EMACx_MR0 defines the operating modes of the EMAC, which can be changed at any time during EMAC operation.

Figure 27-18. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported. **Note:** Be sure than the clock on the PHY side is present If you don't know, do a hard reset or select an internal clock during the soft reset by enabling bit 3 or 4 of SDR0_MFR.
3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

27.7.2 Mode Register 1 (EMACx_MR1)

EMACx_MR1 defines the EMAC operating modes, which can be changed only after a reset.

Software can use EMACx_MR1[FDE] and proper programming of the attached PHY to activate external loop-back mode.

When EMACx_MR1[FDE, ILE] = 1, EMAC wraps transmitted packets back to the receive FIFO without accessing the MII.

Figure 27-19. Mode Register 1 (EMACx_MR1)			
0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MRI[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to *Flow Control* on page 712 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.
4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	In order for the EMAC to process incoming Pause Packets, EMACx_RMR[MAE] and EMACx_RMR[BAE] bits must also be set to 1.
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 1000 Mbps (Gigabit Ethernet mode) 11 Reserved	Defines the possible operational frequency on the MII/GMII interface.
10:12	RFS	Receive (RX) FIFO Size 000 512 bytes 001 1 KB 010 2 KB 011 4 KB 100 8 KB 101 16 KB 110 Reserved	
13:15	TFS	Transmit (TX) FIFO Size 000 512 bytes 001 1 KB 010 2 KB 011 4 KB 100 8 KB 101 16 KB 110 Reserved	
16	TR	Transmit Request 0 Single packet 1 Multiple packets	Defines the different modes for using transmit channel of EMAC.

17:19	MWSW	Maximum Waiting Status Words 000 A packet is not sent until the status of the previous packet has been received 001 Allows up to one status to be pending when sending the next packet 010 Reserved	Defines the number of status words EMAC can wait for, and still continue transmission. A '0' value will force EMAC to wait for every status until requesting a transmission of new packet. For better performance set EMAC0_MR1[MWSW=001]
20	JPSM	Jumbo Packet Support Mode 0 Jumbo packet support mode disabled 1 Jumbo packet support mode enabled	When enabled, EMAC is capable of handling packets with a length of up to 9018 bytes. This bit can be set only when 1000 Mbps mode is chosen
21:25	IPPA	Internal PCS PHY Address	Access to internal PCS/GPCS registers is not supported. Set the IPPA to an address unused by external PHY.
26:28	OBCI	OPB Bus Clock Indication 000 50 MHz 001 66 MHZ 010 83 MHz 011 100 MHz 100 Above 100 MHz	EMACx_MR1[TFS] and data clock of EMACx_MR1 are used for generation of EMC_MDC clock. **Note:** When operational frequency differs from this list, then the next greater frequency should be chosen.
29:31		Reserved	

27.7.3 Transmit Mode Register 0 (EMACx_TMR0)

EMACx_TMR0 defines EMAC operating modes during transmit operations (see *EMAC Transmit Operation* on page 703).

EMACx_TMR0[GNP0, GNP1, GNPD] are self-clearing. Writing 0 to these fields has no effect.

Figure 27-20. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP	Get New Packet 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel	EMACx_TMR0[GNP0] = 0 if EMAC is programmed.
1:28		Reserved	
29:31	TFAE	TX FIFO Almost Empty 000 Reserved 001 Actual number of entries limit in FIFO is 2 and the limit in value of PHY clock cycles is 32 010 Actual number of entries limit in FIFO is 4 and the limit in value of PHY clock cycles is 64 011 Actual number of entries limit in FIFO is 8 and the limit in value of PHY clock cycles is 128 100 Actual number of entries limit in FIFO is 16 and the limit in value of PHY clock cycles is 256 101 Actual number of entries limit in FIFO is 32 and the limit in value of PHY clock cycles is 512 110 Actual number of entries limit in FIFO is 64 and the limit in value of PHY clock cycles is 1024 111 Actual number of entries limit in FIFO is 128 and the limit in value of PHY clock cycles is 2048	When the number of occupied entries in the TX FIFO is less than or equal to this limit, and while a packet is transmitted, TX_FIFO_ALMOST_EMPTY interrupt is asserted. **Note:** The value of '111' is applicable only when 'Jumbo Packet' option is enabled. See *Mode Register 1 (EMACx_MR1)* on page 723. Otherwise, writing a value of '111' will disable the TX FIFO almost empty limit option, that is, no interrupt will occur. This is also set as default option. **Note:** Each FIFO entry is 16 bytes, the width of a FIFO memory array row.

27.7.4 Transmit Mode Register 1 (EMACx_TMR1)

EMACx_TMR1 shown in *Figure 27-21* defines conditions for activation of MAL service requests during transmit operations (see *EMAC Transmit Operation* on page 703).

27.7.4.1 Low-Priority Requests

EMAC requests low priority service from MAL when the number of vacant entries in the transmit FIFO exceeds the decimal transmit low request (TLR) value. EMACx_TMR1[TLR] must be at least 17 when MAL burst length is using the default 64 words.

To avoid a deadlock, the sum of EMACx_TMR1[TLR] and EMACx_TRTR[TRT] must be at least 4 smaller than the transmit FIFO size specified by EMACx_MR1[TFS].

27.7.4.2 Urgent-Priority Requests

EMAC requests urgent priority service from MAL if the following conditions occur:

- EMAC begins transmitting the packet to the media before the entire packet is placed in the TX FIFO
- The number of vacant entries for the currently transmitting packet exceeds the decimal TUR value

Software must coordinate the value of EMACx_TMR1[TUR] with the value of EMACx_MR1[TFS].The value of EMACx_TMR1[TUR] must be smaller than that of EMACx_MR1[TFS] so that the array address encoded in EMACx_TMR1[TUR] can access the full 66-bit wide array.

The binary value of EMACx_TMR1[TUR] must be greater than that of EMACx_TMR1[TLR].

The EMACx_TMR1 contents can be changed only when EMACx_TMR0[GNP0, GNP1, GNPD] = 0.

Figure 27-21. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request	
5:7		Reserved	
8:17	TUR	Transmit Urgent Request	
18:31		Reserved	

27.7.5 Receive Mode Register (EMACx_RMR)

EMACx_RMR defines EMAC operating modes during receive operations.

Figure 27-22. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.	
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.	

2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.	
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.	
4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.	
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.	
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.	
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.	
8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.	
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.	
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.	
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.	
12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.	
13:28		Reserved	

29:31	RFAF	RX FIFO Almost Full 000 Reserved 001 Actual number of entries limit in FIFO is 2 and the limit in value of PHY clock cycles is 32 010 Actual number of entries limit in FIFO is 4 and the limit in value of PHY clock cycles is 64 011 Actual number of entries limit in FIFO is 8 and the limit in value of PHY clock cycles is 128 100 Actual number of entries limit in FIFO is 16 and the limit in value of PHY clock cycles is 256 101 Actual number of entries limit in FIFO is 32 and the limit in value of PHY clock cycles is 512 110 Actual number of entries limit in FIFO is 64 and the limit in value of PHY clock cycles is 1024 111 Actual number of entries limit in FIFO is 128 and the limit in value of PHY clock cycles is 2048	When the number of occupied entries in the RX FIFO is greater than or equal to this limit, and while a packet is received, RX_FIFO_ALMOST_FULL interrupt is asserted. **Note:** The value of '111' is applicable only when 'Jumbo Packet' option is enabled. See *Mode Register 1 (EMACx_MR1)* on page 723. Otherwise, writing a value of '111' will disable the RX FIFO almost full limit option, that is, no interrupt will occur. This is also set as default option. **Note:** Each FIFO entry is 16 bytes, the width of a FIFO memory array row.

27.7.6 Interrupt Status Register (EMACx_ISR)

Each EMAC generates a distinct interrupt event indication. The event indication signal is driven out of the EMAC to UIC1 (interrupt 28 for EMAC0 and interrupt 30 for EMAC1). This interrupt is generated from the content of the EMACx_ISR. The content of the EMACx_ISR is first ANDed with the corresponding mask bits in the EMACx_ISER; the resulting bits are then logically ORed to produce the interrupt signal. Thus, if any of the resulting bits is a 1, an interrupt is generated.

Note: EMAC activates its interrupt signal only after an indication that status for the current packet was accepted by MAL (with the exception of "MMA Operation Succeed/MMA Operation Failed," which causes unconditional activation of interrupt, if it is not masked).

The interrupt indication is cleared by writing 1 to the related bit in the EMACx_ISR; writing 0 has no effect.

The event indication signal is cleared when all non-masked event indication bits are cleared.

Figure 27-23. Interrupt Status Register (EMACx_ISR)

0:1		Reserved	
2	TxPE	TX Parity Error 0 No TX parity error 1 Tx parity error occurs when a transmit data bit is damaged at the transmit FIFO	
3	RxPE	RX Parity Error 0 No RX parity error 1 Rx parity error occurs when a receive data bit is damaged at the receive FIFO or internal LPRA	
4	TxUE	TX Underrun Event 0 No underrun event 1 Underrun event while packet is being transferred	TX FIFO almost empty
5	RxOE	RX Overrun Event 0 No overrun event 1 Overrun event while packet is being received	RX FIFO almost full
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet	

7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet	
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error	
9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than ShortEventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.
13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: • 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) • 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled) • 9018 octets for jumbo packet (with no VLAN support) • 9022 octets for VLAN tagged Jumbo packet
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536). The type field contains a value (1501≤ type ≤ 1535) not defined by IEEE or DIX standards.
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	The type field contains a length (type ≤ 1500) that differs from the length of the received frame.
16:22		Reserved	
23	DB	Dead Bit 0 No transmit error or SQE for TX Channel 1 Transmit error or SQE has occurred for TX Channel	If EMACx_ISR[DB] = 1, EMAC does not request service for TX Channel from MAL, even if EMACx_TMR0[GNP] = 1. EMACx_ISR[DB] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 No SQEs on TX Channel 1 SQE test failure during transmission of a packet from TX Channel	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.

25	TE0	Transmit Error 0 TX Channel transmission OK 1 TX Channel transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
26:29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer not valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

27.7.7 Interrupt Status Enable Register (EMACx_ISER)

EMACx_ISER indicates which conditions in the EMACx_ISR can generate an interrupt.

Each masking bit in the EMACx_ISER corresponds to a related bit in the EMACx_ISR. If a mask bit is set to 1, the corresponding status bit, when set, causes an interrupt to be generated. Setting a mask bit to 0 suppresses interrupt generation for the associated condition.

Mask bits for reserved bits in the EMACx_ISR are not implemented, have no effect on write, and return 0 on read.

Figure 27-24. Interrupt Status Enable Register (EMACx_ISER)

0:1		Reserved	
2	TxPE	TX Parity Error 0 No TX parity error 1 Tx parity error occurs when a transmit data bit is damaged at the transmit FIFO	
3	RxPE	RX Parity Error 0 No RX parity error 1 Rx parity error occurs when a receive data bit is damaged at the receive FIFO or internal LPRA	
4	TxUE	TX Underrun Event 0 No underrun event 1 Underrun event while packet is being transferred	TX FIFO almost empty
5	RxOE	RX Overrun Event 0 No overrun event 1 Overrun event while packet is being received	RX FIFO almost full
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.	
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.	

8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.	
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.	
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.	
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.	
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.	
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.	
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.	
15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.	
16:23		Reserved	
24	SQE	SQE Error 0 SQE error on TX Channel will not generate an interrupt. 1 SQE error on TX Channel will generate an interrupt.	
25	TxE	Transmit Error 0 TX error on TX Channel will not generate an interrupt. 1 TX error on TX Channel will generate an interrupt.	
26:29		Reserved	
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.	

31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.	

27.7.8 Individual Address High (EMACx_IAHR)

EMACx_IAHR contains the high-order halfword of the station unique individual address.

During packet reception, if EMAC is programmed in individual address match mode (EMACx_RMR[IAE] = 1), the contents of EMACx_IAHR are concatenated with the content of EMACx_IALR to form a composite address that is compared with the destination address of the received packet. If addresses match, the packet is transferred to MAL.

During packet transmission, EMACx_IAHR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

Figure 27-25. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31	HO	Receive and Transmit High Order High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

27.7.9 Individual Address Low (EMACx_IALR)

EMACx_IALR contains the low-order word of the station unique individual address.

During packet reception, EMACx_IALR is compared with the corresponding address bits of the received packet.

During packet transmission, EMACx_IALR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

Figure 27-26. Individual Address Low Register (EMACx_IALR)

0:31	LO	Receive and Transmit Low Order Low-order bits of Receive Individual Address or Transmit Source Address	

27.7.10 VLAN TPID Register (EMACx_VTPID)

EMACx_VTPID contains the value of the VLAN TPID (Tag Protocol Identifier) field.

During packet reception, packet bytes 13 and 14 are compared to the content of this register to check whether the packet is tagged with a VLAN ID.

During packet transmission, EMAC uses EMACx_VTPID when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

The value of this register must be a Type field (8100).

Figure 27-27. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved	
16:31	VIDT	VLAN ID tag	

27.7.11 VLAN TCI Register (EMACx_VTCI)

EMACx_VTCI contains the value of the VLAN TCI (Tag Control Information) field.

During packet transmission, EMAC uses EMACx_VTCI when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

Figure 27-28. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved	
16:31	VTCI	VLAN TCI tag	

27.7.12 Pause Timer Register (EMACx_PTR)

EMACx_PTR defines the time period for which the pause function is enabled. EMAC uses EMACx_PTR[TVR] as the timer value field of control (pause) packets (see *Control Packet Transmission* on page 713). Each bit corresponds to 512 bit times.

Figure 27-29. Pause Timer Register (EMACx_PTR)

0:15		Reserved	
16:31	TVF	Timer Value Field	

27.7.13 Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

These registers are used in the hash table function of the multiple individual addressing mode.

See *Address Match Mechanism* on page 717 for more information. See *Figure 27-17* on page 720 for bit mapping information.

Figure 27-30. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved	
16:31	IAHN	Individual Address Hash Number	

27.7.14 Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

These registers are used in the hash table function of the group addressing mode.

See *Address Match Mechanism* on page 717 for more information. See *Figure 27-17* on page 720 for bit mapping information.

Figure 27-31. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved	
16:31	GAHN	Group Address Hash Number	

27.7.15 Last Source Address High (EMACx_LSAH)

EMACx_LSAH contains the high-order halfword of the source address of the last "good" received packet. The packet is considered to be "good" if EMAC is programmed to provide this packet to MAL.

Figure 27-32. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved	
16:31	LSAHHW	Last Source Address High-Order Halfword	

27.7.16 Last Source Address Low (EMACx_LSAL)

EMACx_LSAL contains the low-order word of the source address of the last "good" received packet. The packet is considered "good" if EMAC is programmed to provide this packet to MAL.

Figure 27-33. Last Source Address Low Register (EMACx_LSAL)

0:31	LSALW	Last Source Address Low-Order Word	

27.7.17 Inter-Packet Gap Value Register (EMACx_IPGVR)

EMACx_IPGVR contains the value of one-third of the inter-packet gap (IPG) for the next packet to be transmitted. ("Frame" is synonymous with "packet.")

The resolution of each bit is 8-bit times. The minimum value in the register is four, causing a minimum.

Figure 27-34. inter-Packet Gap Value Register (EMACx_IPGVR)

0:25		Reserved	
26:31	IPG	Inter-Packet Gap	

27.7.18 STA Control Register (EMACx_STACR)

EMACx_STACR controls the MII/GMII Management interface. The software must follow the following steps during access to the EMACx_STACR:

1. EMAC sets EMACx_STACR[OC] = 0 to indicate that the data has been written to the PHY, or the data read from the PHY is valid. The device driver should poll for EMACx_STACR[OC] = 0 before issuing a new command, or before using data read from the PHY.
2. Software writes the EMACx_STACR register fields PHYD, IMS, STAOPC, PADR, and DEVADR as needed for the desired command.
3. Software sets EMACx_STACR[OC] = 1 to start the transaction.
4. Return to step 1.

Figure 27-35. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	MMA Go Bit/Operation Complete 0 Operation completed 1 Transaction started	When set to 1 (by software), the transaction starts. This is a self-clearing bit. EMAC clears this bit when the operation completes (either when the data is written to the PHY or when the data read from the PHY is valid). The device driver should poll this bit for a 0 before issuing a new command or before using data read from the PHY. This bit is set to 0 after a POR, after a soft reset, or when a new transaction is started (software triggers the MMA Go Bit). Writing 0 to this bit has no effect.
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	
18	IMS	Indirect Mode Selection 0 Direct access 1 Indirect access	This bit is set to 1 if the access method is Indirect Mode (according to Clause 45 of IEEE802.3). Setting this bit to 0 enforces the use of the old MIO/MDIO (Clause 22 of IEEE 802.3) access method.This bit affects the use of the EMACx_STACR[STAOPC] bit field.

19:20	STAOPC	STA Opcode Direct access OP codes (IMS=0): 01 Write 10 Read Indirect access OP codes (IMS=1): 00 Address 01 Write 10 Read 11 Read Inc.	This field contains the opcode to be used during the MIO/ MDIO transaction (see IMS field).
21		Reserved	
22:26	PADR	PHY Command Destination Address	This field contains the address of the PHY intended to receive the command. To access an external PHY, set PADR to the desired address and set EMACx_MR1[IPPA] to an address unused by any external PHY.
27:31	DEVADR	PHY Register Address	This field contains the PHY register address.

27.7.19 Transmit Request Threshold Register (EMACx_TRTR)

EMACx_TRTR defines the conditions that cause EMAC to initiate transmission to the Ethernet MAC sub-block, and for requesting service from MAL.

EMACx_TRTR[TRT] defines the number of occupied entries in the transmit FIFO that should be written before the transmit FIFO control logic initiates a transmit request to the Ethernet MAC sub-block.

If an entire packet is already located in the transmit FIFO, EMAC initiates a transmit regardless of the programmed value.

The software must coordinate the value of EMACx_TRTR[TRT] with the transmit FIFO size specified in EMACx_MR1[TFS].

To avoid deadlock, the sum of EMACx_ TMR1[TLR] and EMACx_TRTR[TRT] must be smaller, by at least 4, than the transmit FIFO specified in EMACx_MR1[TFS].

To avoid an underrun, program this threshold to a high enough value.

In half-duplex mode, in case of collision, to allow packet re-transmission without involving MAL, EMAC preserves the necessary space in the Transmit FIFO unless it gets an indication that the collision window has elapsed.

The EMACx_TRTR can be written only while EMACx_MR0[TXI] = 1.

Figure 27-36. Transmit Request Threshold Register (EMACx_TRTR)			
0:7	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 0000_0000 64 bytes 0000_0001 128 bytes 0000_0010 192 bytes 0000_0011 256 bytes . . . 1111_1100 16,192 bytes 1111_1101 16,256 bytes 1111_1110 16,320 bytes 1111_1111 16,384 bytes	
8:31		Reserved	

27.7.20 Receive Low/High Water Mark Register (EMACx_RWMR)

The EMACx_RWMR defines the conditions that cause the EMAC to activate a low or urgent priority MAL request, and that manage flow control.

EMAC activates a low priority request if the number of occupied entries in the receive FIFO is greater than or equal to the content of EMACx_RWMR[RLWM] (the receive low water mark is reached). A request for a pause packet with a pause_value of 0 is also issued when the receive low water mark is reached.

Software must coordinate the value of EMACx_RWMR[RLWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RLWM] should be smaller than EMACx_MR1[RFS] and larger than the MAL burst length.

Note: In the PPC440EPx/GRx, the user can select the MAL burst length for each MAL RX or TX channel by writing to the appropriate fields in SDR0_MALRBL and SDR0_MALTBL registers described in *MAL Receive Burst Length Register (SDR0_MALRBL)* on page 651 and *MAL Transmit Burst Length Register (SDR0_MALTBL)* on page 652.

If the entire packet is already in the receive FIFO, EMAC initiates a low priority request regardless of the programmed value.

EMAC activates an urgent priority request if the number of occupied entries in the Receive FIFO is greater than or equal to EMACx_RWMR[RHWM] (the receive high water mark is reached). A request for a pause packet is also issued when the receive high water mark is reached.

Software must coordinate the value of EMACx_RWMR[RHWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RHWM] should be greater than the value of EMACx_RWMR[RLWM] and less then the size of the receive FIFO.

Figure 27-37. Receive Low/High Water Mark Register (EMACx_RWMR)

0:9	RLWM	Receive Low Water Mark	
10:15		Reserved	
16:25	RHWM	Receive High Water Mark	
26:31		Reserved	

27.7.21 Transmitted Octects (EMACx_OCTX)

The read-only EMACx_OCTX register contains the number of transmitted octets.

Figure 27-38. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.	

27.7.22 Received Octects Register (EMACx_OCRX)

The read-only EMACx_OCRX register contains the number of received octets.

Figure 27-39. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.	

27.8 MII/GMII Interface

EMAC implements all MII functionality in accordance with Clause 22 in the IEEE Std. 802.3u (when operating in 10/100 Mbps media) and all GMII interface functionality in accordance with Clause 35 in the IEEE Std. 802.3z (when operating in 1000 Mbps media).

The MII/GMII interface is a reconciliation sublayer interface which allows a variety of PHYs to be attached to the EMAC Ethernet MAC without future upgrade problems.

27.8.1 MII Station Management Interface

The EMAC MII station management unit (STA) implements a specific protocol and a special packet format to exchange management packets with the registers of the attached PHY. EMAC automatically generates MII management packets, which conform to Clause 22 in IEEE Std. 802.3u. EMAC uses the EMACx_STACR for generation of the management packet. illustrates the interface.

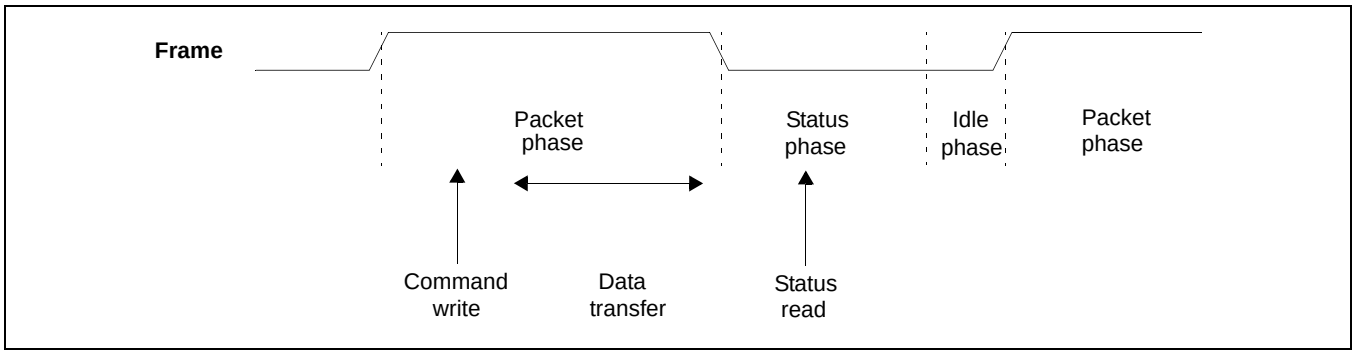
27.9 MAL – EMAC Packet Transfer Flow

The packet transfer flow consists of three phases. These three phases are used to define the details of the EMAC-MAL protocol.

1. Packet phase - EMAC initiates a packet transfer operation. The packet transfer is started by a command write. During command write MAL provides control information for EMAC on a per-packet basis. Following the command write, MAL begins the data transfer, during which MAL transfers data between the buffers located in the system's memory and EMAC. In transmit, the data is transferred from the system's memory to EMAC, while in receive, the data is transferred from EMAC to the system's memory buffers.
   - EMAC remains in the packet phase until the data transfer has been completed or a ready status can be returned to MAL. The packet phase ends when EMAC deasserts the FRAME signal associated with the related channel (receive/transmit).
   - The packet phase is defined by activity of an appropriate FRAME signal.
2. Status phase - This is the second phase of the packet transfer. Following the de-assertion of the FRAME signal, EMAC switches to the status phase. At this stage, EMAC uses an appropriate signal as a request for service which is interpreted by MAL as a request for status read.
3. Idle phase - EMAC moves into the idle phase following a reset or after status was transferred (end of status phase). During the idle phase, EMAC cannot send any signals to MAL, nor can MAL send any active signals to EMAC. EMAC exits the idle phase by asserting the FRAME signal (and entering the packet phase described above). Idle phase can be skipped when EMAC operates in multiple transfer mode.

Figure 27-40 illustrates the different phases in the EMAC-MAL communication.

Figure 27-40. EMAC-MAL Communication Phases



During the packet and status phases EMAC signals a request for service by driving its arbitration level signal to a non-idle level.

27.10 Programming Notes

Certain combinations in device drivers are not allowed when writing to EMAC registers. When creating device drivers, ensure that the following guidelines are used:

- In half-duplex mode (EMACx_MR1[FDE] = 0) while working in Gigabit Ethernet mode 1000 Mbps (EMACx_MR1[MF = 10]), the transmit FIFO size (EMACx_MR1[TFS] must be greater than 512 bytes.
- When internal loopback is enabled (EMACx_MR1[ILE] = 1), EMAC must be configured in full-duplex mode (EMACx_MR1[FDE] =1)
- EMACx_MR1[IST] = 0 only when EMACx_MR1[MF] = 0:10 and EMACx_MR1[FDE] =0
- EMACx_MR1[EIFC] = 0 if EMACx_MR1[FDE] =0

- EMACx_TMR1[TLR] must be greater than the MAL burst size in entities (6 for MAL)
- EMACx_TMR1[TUR] must be greater or equal to EMACx_TMR1[TLR] and less than the Transmit FIFO size in entries (EMACx_MR1[TFS])
- To avoid deadlock, the sum of EMACx_TMR1[TLR] and the EMACx_TRTR[TRT] must be at least four less than the Transmit FIFO size specified in EMACx_MR1[TFS]
- EMACx_RWMR[RLWM] must be greater than the MAL burst size in entities (six for MAL)
- EMACx_RWMR[RHWM] must be greater than EMACx_RWMR[RLWM]
- EMACx_RWMR[RHWM] must be less than the Receive FIFO size in entities (EMACx_MR1[RFS])

27.10.1 Reset and Initialization

The EMAC must be initialized after a reset, or before performing configuration changes. The following types of reset operations can be applied to EMAC.

- Hard Reset. When RESET input is asserted, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. To be recognized, the reset signal must be asserted for at least two cycles of the slowest clock domain inside EMAC (indicating that the hard reset must be at least 800 ns).
- Soft Reset. Software first should reset the appropriate MAL channels and then begin a soft reset by setting EMACx_MR0[SRST] = 1. In response to the soft reset, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. After EMAC finishes all activities related to the soft reset processing, EMACx_MR0[SRST] = 0.
- Smart Reset. The software initializes smart reset mode by writing 0 to EMACx_MR0[TXE] or EMACx_MR0[RXE], or to both. In this case, the Ethernet MAC sub-block completes on-going activity (receive, transmit, or both) and then goes to the related Idle state (indicated by setting either EMACx_MR0[TXI] = 1 or EMACx_MR0[RXI] = 1, or both). In this case, the control logic sub-block of EMAC is still accessible for OPB and MAL transactions.

Before performing the necessary configuration changes in EMAC, the software must follow one of the following scenarios. Then the EMAC can be properly configured.

27.10.1.1 Scenario 1

- Hard/soft reset was activated.
- During hard/soft reset, EMACx_MR0[TXE] and EMACx_MR0[RXE] are reset.
- Software detects that the EMACx_MR0[SRST] is reset (after soft reset only).
- Software keeps EMACx_TMR0[GNP] = 0.
- The software can change one or more fields in registers marked with a Reset write access mode in *Table 27-6 EMAC0 Register Summary* on page 720 (actually, all EMAC registers are accessible in this scenario).
- The software initializes EMACx_TMR0[GNP] as appropriate.
- The software configures EMACx_MR0[TXE, RXE].

27.10.1.2 Scenario 2

- Software sets EMACx_MR1[EIFC] = 0.
- Software sets EMACx_MR0[TXE] = 0.
- The TXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[TXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[TXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[TXE] = 0. The software can access only part of the EMAC registers marked with write access mode T in *Table 27-6 EMAC0 Register Summary* on page 720.
- After all configuration is done, software can set EMACx_MR0[TXE] = 1.

Note: When Scenario 2 occurs, EMAC can still receive packets if EMACx_MR0[RXE] = 1. Scenarios 2 and 3 can occur simultaneously.

27.10.1.3 Scenario 3

- Software sets EMACx_MR0[RXE] = 0.
- The RXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[RXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[RXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[RXE] = 0. The software can access only part of EMAC registers marked with write access mode R in *Table 27-6 EMAC0 Register Summary* on page 720.
- After all configuration is done, software can set EMACx_MR0[RXE] = 1.

Note: When Scenario 3 occurs, EMAC can still transmit packets if EMACx_MR0[TXE] = 1. Scenarios 2 and 3 can occur simultaneously.

28. Serial Port Operations

The PPC440EPx/GRx contains four universal asynchronous receiver/transmitters (UARTs) to support communications with serial peripheral devices. Each UART includes a 16-byte send and a 16-byte receive FIFO.

Features of the UART include:

- 16-byte send FIFO, 16-byte receive FIFO
- Full duplex operation
- Programmable baud rate generator
- Supports 5- to 8-bit word size, 1 or 2 stop bits, even, odd, or no parity
- One 8-wire interface, two 4-wire interfaces, one 4-wire interface and two 2-wire interfaces, or four 2-wire interfaces
- Hardware flow control is selectable

The UART performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions, such as parity, overrun, framing, and break interrupt.

This UART powers up in character mode and can be put into FIFO mode to relieve the processor of excessive software overhead. Here, internal FIFOs are activated allowing 16 bytes (plus 3 bits per byte of error data in the RCVR FIFO) to be stored in both receive and transmit modes.

The source of the UART serial clock input is selected by SDR0_UART0:3[U0EC:U3EC]. Either the internal serial clock or an external serial clock can be selected. A programmable baud rate generator is included that is capable of dividing the UART serial clock input by a divisor of 1 to (2^{16} – 1) and producing the 16× clock required for driving the UART internal transmitter/receiver logic. The internal serial clock input is derived from the PLB clock by a divisor specified by SDR0_UARTx[UxDIV]. See *Clocking* on page 273 for additional information.

The UART has an interrupt system that can be programmed to the user's requirements, helping to minimize the computing required to handle the communications link. UART interrupts are capable of triggering an interrupt request to the PPC440EPx/GRx interrupt controller.

28.1 Functional Description

- Equipped with complete status reporting capability
- Transmitter and receiver are each buffered with 16-byte FIFOs when FIFO mode selected
- Can add/delete standard asynchronous communication bits such as start, stop, and parity to/from the serial data
- When in character mode, holding and shift registers eliminate the need for precise synchronization between the processor and serial data
- Full prioritized interrupt system controls
- Independently controlled transmit, receive, line status, and data set interrupts
- Programmable baud rate generator divides the UART serial clock input by 1 to (2^{16}–1) and generates the 16x clock:

  Baud rate (bps) = (Serial Clock Input) / (16 x Decimal Divisor)

- Receiver uses 5-way oversampling as follows: it samples each serial bit five times, and if at least three of the samples are 1's, the bit is determined to be a 1, otherwise it is a 0
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no parity bit generation and detection
  - 1-,1.5-, or 2-stop bit generation
  - Variable baud rate
- Line break generation and detection, and false start bit detection
- Internal diagnostic capability:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation

28.2 Serial Port Clocking

Each of the four PPC440EPx/GRx UARTs can be clocked individually from an external serial clock or from a single internally generated serial clock. The serial clock source is selected by the SDR0_UARTx[UxIC, UxEC] bit fields. The internally generated serial clock is derived from the PLB clock, and can only be an integer (n) fraction of that clock where n is in the range from 1 to 256.

Formula for the Serial Clock and Baud Rate when using the PLB as the clock source:

PLB:UART Divisor = SDR0_UART0[UxDIV]

Serial Clock = PLBClk/PLB:UART Divisor

UART Divisor = UARTx_DLM | UARTx_DLL

BAUD Rate = Serial Clock / (16* UART Divisor)

The choice of serial clock frequency affects the serial communications error rate. If an external clock of 1.8432 MHz (or some multiple of this frequency) is used, the error rate approaches zero. However the internally-generated clock operates only at certain clock frequencies, all of which result in a small, non-zero error rate.

Formula for the Serial Clock and Baud Rate when using the external serial clock, UARTSerClk, as the clock source:

Serial Clock = UARTSerClk

UART Divisor = UARTx_DLM | UARTx_DLL

BAUD Rate = Serial Clock / (16* UART Divisor)

Acceptable baud rates are always integral multiples of 300 (for example, 1200 = 4 × 300). *Table 28-1* shows optimum UART divisor and divide ratios for a range of possible baud rates. This information is provided for various clock frequencies. Note that the serial clock frequency must be less than half the OPB frequency. The UART divisor is programmed in UARTx_DLM and UARTx_DLL (see *Divisor Latch LSB and MSB Registers (UARTx_DLL and UARTx_DLM)* on page 753). The value range is 1 to (2^16-1)=65535.

Table 28-1. Baud Rate Settings

Desired Baud Rate (bps)	PLB:UART Divide Ratio	Serial Clock Frequency (MHz)	Minimum OPB Frequency (MHZ)	UART Divisor	Actual Baud Rate (bps)	Error (%)
PLB Clock =133 MHz						
1200	14	9.5238	19.4175	496	1200.07	0.006
2400	14	9.5238	19.4175	248	2400.15	0.01
4800	14	9.5238	19.4175	124	4800.31	0.01
9600	14	9.5238	19.4175	62	9600.61	0.01
19200	14	9.5238	19.4175	31	19201.23	0.01
28800	17	7.8431	15.0376	17	28835.06	0.12
33600	31	4.3011	8.4033	8	33602.15	0.01
38400	31	4.3011	8.4033	7	38402.46	0.01
57600	29	4.5977	9.2807	5	57471.26	-0.22
115200	24	5.5556	11.2359	3	115740.74	0.47
307200	27	4.9383	9.9751	1	308641.97	0.47
PLB Clock = 166.66 MHz						
1200	31	5.3763	10.8695	280	1200.08	0.01
2400	31	5.3763	10.8695	140	2400.15	0.01
4800	31	5.3763	10.8695	70	4800.31	0.01
9600	31	5.3763	10.8695	35	9600.61	0.01
19200	32	5.2083	10.5263	17	19148.28	-0.27
28800	19	8.7719	17.8571	19	28855.03	0.19
33600	31	5.3763	10.8695	10	33602.15	0.01
38400	17	9.8039	20.0000	16	38296.57	-0.27
57600	30	5.5556	11.2359	6	57870.37	0.47
115200	30	5.5556	11.2359	3	115740.74	0.47
307200	17	9.8039	20.0000	2	306372.55	-0.27

28.3 UART Registers

UART registers are accessed via memory to 0x1 EF60_0XYY where X=3 for UART0, X=4 for UART1, X=5 for UART2 and X=6 for UART3.

Table 28-2. UART Configuration Registers

Register	Address	Access	Description	Page
UARTx_RBR	0x1 EF60_0X00 [1]	R	UART x Receiver Buffer Register	747
UARTx_THR	0x1 EF60_0X00 [1]	W	UART x Transmitter Holding Register	747
UARTx_IER	0x1 EF60_0X01 [1]	R/W	UART x Interrupt Enable Register	747
UARTx_IIR	0x1 EF60_0X02	R	UART x Interrupt Identification Register	748
UARTx_FCR	0x1 EF60_0X02	W	UART x FIFO Control Register	749
UARTx_LCR	0x1 EF60_0X03	R/W	UART x Line Control Register	749
UARTx_MCR	0x1 EF60_0X04	R/W	UART x Modem Control Register	750
UARTx_LSR	0x1 EF60_0X05	R/W	UART x Line Status Register	751
UARTx_MSR	0x1 EF60_0X06	R/W	UART x Modem Status Register	752
UARTx_SCR	0x1 EF60_0X07	R/W	UART x Scratch Register	753
UARTx_DLL	0x1 EF60_0X00 [1]	R/W	UART x Divisor Latch (LSB)	753
UARTx_DLM	0x1 EF60_0X01 [1]	R/W	UART x Divisor Latch (MSB)	753

Note 1: UARTx_LCR[DLAB] controls the function accessed through registers 0x1 EF60_0X00 and 0x1 EF60_0X01. When UARTx_LCR[DLAB] is 0, access is enabled to the Receiver/Transmitter registers and the Interrupt Enable register. When UARTx_LCR[DLAB] is a 1, access is enabled to the Divisor Latch registers.

The system programmer can access any of the UART registers via the processor. These registers control all UART operations including transmission and reception of data. In the 440EPx there are four UARTs, designated 0 through 3. In the following sections, the registers are specified with a generic name where x represents 0, 1, 2 or 3. For example, the Line Control Register appears as a UARTx_LCR.

In the four wire interfaces, two of the four wires are TX and RX. The remaining two wires can be programmed as a combination of DTR and DSR, or CTS and RTS. The selection is made in SDR0_PFC1[U0ME] and SDR0_PFC1[U1ME] respectively. DCD and RI are not available on the 4-wire interface.

28.3.1 Receiver Buffer Registers (UARTx_RBR)

Figure 28-1. UART Receiver Buffer Registers (UARTx_RBR)			
0:7		Data bit	
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.2 Transmitter Holding Registers (UARTx_THR)

Figure 28-2. UART Transmitter Holding Registers (UARTx_THR)			
0:7		Data bit	
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.3 Interrupt Enable Registers (UARTx_IER)

Five UART interrupts on four priority levels are enabled via the Interrupt Enable Register, UARTx_IER. Any of the five interrupts can be used to surface a UART interrupt to the PPC440EPx/GRx interrupt controller. Each interrupt can be enabled by setting its appropriate bit. Resetting UARTx_IER[4:7] totally disables the UART interrupt system. Disabling an interrupt prevents it from being shown as active in the UARTx_IIR and prevents it from signaling a UART interrupt to the PPC440EPx/GRx interrupt controller. See *Table 28-3 Interrupt Priority Level* on page 748.

Figure 28-3. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/ disable state of ERBFI.
Note: UART0_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.4 Interrupt Identification Registers (UARTx_IIR)

The UART prioritizes interrupts into four levels which are recorded in the Interrupt Identification Register. The interrupt types in the order of their priority are as follows:

1. Receiver line status
2. Received data available and character timeout indication
3. Transmitter holding register empty
4. Modem status

Table 28-3. Interrupt Priority Level

IIR Bit 4	IIR Bit 5	IIR Bit 6	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	1	Receiver Line Status	Overrun, Parity or Framing Error, or Break Interrupt.	Read LSR.
0	1	0	2	Received Data Available	Receiver data available or trigger level reached.	Read RBR, or FIFO drops below trigger level.
1	1	0	2	Character Timeout Indication	No characters have been removed from or input to the receiver FIFO during the last four character times and it contains at least one character during this time.	Read RBR.
0	0	1	3	Transmitter Holding Register Empty	Transmitter Holding Register Empty.	Read IIR (if source of interrupt) or write THR.
0	0	0	4	Modem Status	Clear to Send, Data Set Ready, Ring Indicator or Data Carrier Detect.	Read MSR.

When the processor accesses UARTx_IIR, the UART records new interrupts, but does not change its current contents until the access by the processor is complete. The UART indicates the highest priority interrupt pending to the PPC440EPx/GRx interrupt controller via the IIR.

Figure 28-4. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FE	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)	
2:3		Reserved	
4:6	INTID	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved	**Note:** Priority 1 is highest priority.

7	INTP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending	When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.5 FIFO Control Registers (UARTx_FCR)

The FIFO control register has the same address as the IIR and is a write-only register. This register is used to perform FIFO control operations such as selecting the type of DMA signaling, setting the receiver FIFO trigger levels, clearing the FIFOs, and enabling the FIFO.

Figure 28-5. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 04 bytes 10 08 bytes 11 14 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.6 Line Control Registers (UARTx_LCR)

The system programmer uses the line control register (LCR) to specify the format of the asynchronous data communications exchange and to set the Divisor Latch Access bit. The contents of the LCR can also be read by the processor. The read capability simplifies system programming, and eliminates the need for separate storage of the line characteristics in system memory.

Figure 28-6. UART Line Control Registers (UARTx_LCR)			
0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR [EPS] = 0 and UARTx_LCR[PEN] = 1,the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
Note: UART0_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.7 Modem Control Registers (UARTx_MCR)

The interface between the modem, data set, or peripheral device emulating a modem, and the UART, is controlled by the Modem Control Register (UARTx_MCR).

Figure 28-7. UART Modem Control Registers (UARTx_MCR)			
0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	

3	LOOP	Loopback Mode 0 Disabled 1 Enabled	Provides a local loop back feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{DSR}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DCD}$ are disconnected. 4. The four modem control outputs $\overline{DTR}$, $\overline{RTS}$, $\overline{OUT1}$, and $\overline{OUT2}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.
4	OUT2	User Output 2 0 $\overline{OUT2}$ inactive (1) 1 $\overline{OUT2}$ active (0)	The $\overline{OUT2}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{OUT1}$ inactive (1) 1 $\overline{OUT1}$ active (0)	The $\overline{OUT1}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 $\overline{RTS}$ inactive (1) 1 $\overline{RTS}$ active (0)	
7	DTR	Data Terminal Ready 0 $\overline{DTR}$ inactive (1) 1 $\overline{DTR}$ active (0)	
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.8 Line Status Registers (UARTx_LSR)

Information concerning the data transfer is held for the processor in this register. Bits 3 through 6 are conditions that produce a receiver line status interrupt whenever the condition corresponding to the active bit is detected and the interrupt is enabled. This register is intended for read operations only and writing is not recommended.

Figure 28-8. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character. 1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.	

2	THRE	Transmitter Holding Register Empty Indicator 0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO. 1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the PPC440EPx/GRx interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.
3	BI	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	Framing Error Indicator. 0 Reset to 0 whenever processor reads LSR. 1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level). Indicates that a valid stop bit was not found in the received character.	Error causes a Receiver Line Status Interrupt.
5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR.[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.
6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.9 Modem Status Registers (UARTx_MSR)

The processor can monitor the present state of the modem (or peripheral device) control lines by reading the Modem Status Register (UARTx_MSR). In addition, the UARTx_MSR has four bits to indicate if any of the modem (or peripheral device) control lines have changed state.

Figure 28-9. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].

2	DSR	Data Set Ready	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loop back mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{DCD}$ input changed state	Indicates that the $\overline{DCD}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{RI}$ input changed from 0 to 1	Indicates that the $\overline{RI}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 $\overline{DSR}$ input changed state	Indicates that the $\overline{DSR}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{CTS}$ input changed state	Indicates that the $\overline{CTS}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.10 Scratchpad Registers (UARTx_SCR)

A scratchpad register intended for use by the programmer as a temporary data location is provided in this UART. It does not control the UART operation in any way.

Figure 28-10. UART Scratchpad Registers (UARTx_SCR)

0:7		Data bits	
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

28.3.11 Divisor Latch LSB and MSB Registers (UARTx_DLL and UARTx_DLM)

The divisor latches are used to program the UART divisor used in generating the baud clock. A 16-bit divisor may be programmed through these registers. Access to these registers is provided by setting UARTx_LCR[DLAB] = 1. These registers have a power-on reset value of 0.

Figure 28-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits	
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

Figure 28-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits	
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the lsb.			

The UART divisor is calculated using the following formula:

UART Divisor = Serial Input Clock/(16 x Baud Rate)

For example, if the serial input clock= 11.0592MHz and a baud rate of 9600 bps is required:

UART Divisor = Serial Input Clock/(16 x Baud Rate)

= 11,059,200/(16x9600)

= 72 = 0x48

For this example, UARTx_DLM should be programmed to 0 and UARTx_DLL register should be programmed to 0x48. Due to the error introduced by rounding, some baud rates cannot be generated at certain serial input clock frequencies. *Table 28-4* lists some common baud rates and their corresponding Divisor Latch register values with a serial input clock of 11.0592MHz.

Table 28-4. Divisor Latch Settings for Certain Baud Rates

Baud Rate (bps)	Divisor Latch MSB	Divisor Latch LSB
9600	0x00	0x48
19200	0x00	0x24
28800	0x00	0x18
38400	0x00	0x12
57600	0x00	0x0C

28.3.12 UART Configuration Register 0 (SDR0_UART0)

Figure 28-13 describes SDR0_UART0 register bits.

Reset value = 0x20000026

Figure 28-13. UART Configuration Register 0 (SDR0_UART0)

0:3	U0ICS	UART 0 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART 0 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock	
9	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled	
10	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled	

11	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U0DIV	UART 0 Divider 0000_0000 - divider = 256 0000_0001 - divider = 1 0000_0010 - divider = 2 ..... 1111_1111 - divider = 255	The source for SDR0_UART0[U0DIV] is indicated by SDR0_UART0[U0ICS]

28.3.13 UART Configuration Register 1 (SDR0_UART1)

Figure 28-14 describes SDR0_UART1 register bits.

Reset value = 0x20000026

Figure 28-14. UART Configuration Register 1 (SDR0_UART1)

0:3	U1ICS	UART 1 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART1 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock	
9	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled	
10	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled	
11	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U1DIV	UART 1 Divider 0000_0000 - UART1 divisor = 256 0000_0001 - UART1 divisor = 1 0000_0010 - UART1 divisor = 2 ..... 1111_1111 - UART1 divisor = 255	The source for SDR0_UART1[U1DIV] is indicated by SDR0_UART1[U1ICS]

28.3.14 UART Configuration Register 2 (SDR0_UART2)

Figure 28-15 describes SDR0_UART2 register bits.

Reset value = 0x20000026

Figure 28-15. UART Configuration Register 2 (SDR0_UART2)

0:3	U2ICS	UART 2 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART2 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U2EC	UART 2 EXT Clock Enable 0 UART2 uses the internal serial clock 1 UART2 uses the external serial clock	
9	U2DTE	UART 2 DMA Transmit Enable 0 UART2 DMA transmit is disabled 1 UART2 DMA transmit is enabled	
10	U2DRE	UART 2 DMA Receive Enable 0 UART2 DMA receive is disabled 1 UART2 DMA receive is enabled	
11	U2DC	UART 2 DMA Clear 0 U2DTE and U2DRE are not cleared when UART receives a corresponding terminal count. 1 U2DTE and U2DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U2DIV	UART 2 Divider 0000_0000 - UART2 divisor = 256 0000_0001 - UART2 divisor = 1 0000_0010 - UART2 divisor = 2 ..... 1111_1111 - UART2 divisor = 255	The source for SDR0_UART2[U2DIV] is indicated by SDR0_UART2[U2ICS]

28.3.15 UART Configuration Register 3 (SDR0_UART3)

Figure 28-16 describes SDR0_UART3 register bits.

Reset value = 0x20000026

Figure 28-16. UART Configuration Register 3 (SDR0_UART3)

0:3	U3ICS	UART 3 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART3 internal clock source = PLB 0011 Reserved	
4:7		Reserved	
8	U3EC	UART 3 EXT Clock Enable 0 UART3 uses the internal serial clock 1 UART3 uses the external serial clock	

9	U3DTE	UART 3 DMA Transmit Enable 0 UART3 DMA transmit is disabled 1 UART3 DMA transmit is enabled	
10	U3DRE	UART 3 DMA Receive Enable 0 UART3 DMA receive is disabled 1 UART3 DMA receive is enabled	
11	U3DC	UART 3 DMA Clear 0 U3DTE and U3DRE are not cleared when UART receives a corresponding terminal count. 1 U3DTE and U3DRE are cleared when UART receives a corresponding terminal count.	
12:23		Reserved	
24:31	U3DIV	UART 3 Divider 0000_0000 - UART3 divisor = 256 0000_0001 - UART3 divisor = 1 0000_0010 - UART3 divisor = 2 ..... 1111_1111 - UART3 divisor = 255	The source for SDR0_UART3[U3DIV] is indicated by SDR0_UART3[U3ICS]

28.4 FIFO Operation

This section discusses the various modes of operation including interrupt mode, polled mode, and sleep mode.

28.4.1 Interrupt Mode

The following sections describe the operation of the UART in interrupt mode.

28.4.1.1 Receiver

Receiver interrupts occur as described below when the receiver FIFO and receiver interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ERBFI] = 1.

The received data available interrupt is issued when the number of characters in the FIFO has reached the trigger level programmed into UARTx_FCR. This interrupt is reset to 0 when the FIFO character count drops below this trigger level.

The receiver line status interrupt (UARTx_IIR = 0xC6) is a top priority interrupt, whereas the received data available interrupt (UARTx_IIR = 0xC4) is a second priority interrupt.

Data Ready (UARTx_LSR[DR]) is set as soon as a character is transferred from the shift register to the receiver FIFO. This bit is reset when the FIFO is empty.

Receiver timeout interrupts will occur as described below when the receiver FIFO and receiver interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ERBFI] = 1.

A FIFO timeout will occur when:

At least one character is in the receiver FIFO, no serial characters have been received for four serial character time periods, and the processor has not read the FIFO for four serial character time periods. A serial character time period is as follows:

1/(baud rate) x (# start bits + word length + # parity bits + # stop bits)

For example, the serial character time period for an 8-bit word with one parity bit, two stop bits at 56K baud is as follows:

1/(56000) x (1 + 8 + 1 + 2) = 214.3µs

So the timeout would occur after 857.1µs, if the above conditions hold.

When a timeout interrupt has occurred, it is cleared and the timer is reset when the processor reads one character from the receiver FIFO.

When a timeout interrupt has not occurred, the timer is reset after a new serial character is received or the processor reads the receiver FIFO.

28.4.1.2 Transmitter

Transmitter interrupts occur, as described below, when the transmitter FIFO and transmitter interrupts are enabled by setting UARTx_FCR[FE] = 1 and UARTx_IER[ETBEI] = 1.

The transmitter holding register interrupt (UARTx_IIR = 0xC2) occurs when transmit FIFO is empty, and is cleared as soon as the transmitter holding register is written to or the IIR is read. One to 16 characters may be written to the transmitter FIFO while servicing this interrupt.

The transmitter FIFO empty indications are delayed by one character time minus the last stop bit time whenever the following event occurs: UARTx_LSR[THRE] = 1 and there were less than two bytes simultaneously present in the transmit FIFO since the last UARTx_LSR[THRE] = 1. If UARTx_FCR[FE] = 1 (FIFOs enabled), the first transmitter interrupt after changing UARTx_FCR[FE] is immediate.

Receiver FIFO trigger level interrupts, received data available interrupts, and character timeouts all have equivalent second interrupt priority. Current transmitter holding register empty interrupt and Transmit FIFO empty have equivalent third interrupt priority.

28.4.2 Polled Mode

When UARTx_FCR[FE] = 1 (FIFOs enabled), and UARTx_IER[5:7] are all set to 0 (interrupts disabled), the UART is in FIFO polled mode of operation. The receiver and transmitter are controlled separately, so either can be in polled mode of operation. In polled mode, the user program must check the UARTx_LSR to see the status of the receiver and/or transmitter.

UARTx_LSR[3:6] specifies which errors (if any) have occurred. Character status errors are handled in the same way as in interrupt mode. Since UARTx_IER[ELSI] = 0, the IIR is not affected. UARTx_LSR[DR] is set as long as there is at least one character in the receiver FIFO. UARTx_LSR[THRE] indicates if the transmitter FIFO is empty. UARTx_LSR[TEMT] indicates if the transmitter FIFO and the transmitter shift register are empty. UARTx_LSR[RFE] indicates if there are any errors in the receiver FIFO.

In FIFO polled mode, there are no character timeout or trigger levels; however, the FIFOs are still capable of holding characters.

28.4.3 UART and Sleep Mode

All UARTs can be placed in sleep mode via the UART sleep bits in the CPM0_ER register (CPM0_ER[UART0:UART1:UART2_UART3]). The most common usage would be to save a little power if one or any of the UARTs were not going to be used.

Using sleep mode dynamically requires careful software control to make sure the UARTs are idle before putting them to sleep.

28.5 DMA Operation

The DMA controller can be configured to perform DMA operations using UART, which appears as an 8-bit peripheral to the DMA controller. When selected, the UART receiver is internally wired to the DMAReq and DMAAck signals of DMA channel 2, and the transmitter is internally wired to the DMAReq and DMAAck signals of DMA channel 3.

The UART can be operated in FIFO mode or non-FIFO mode. In FIFO mode, the transfers can be done as single transfers (DMA mode 0) or multiple transfers (DMA mode 1), depending on the setting of the DMS field in the FIFO Control Register (FCR). In non-FIFO mode, DMA transfers are performed using single transfers, using the UART's DMA mode 0. This section describes proper UART DMA programming. For more information on general DMA programming, see *Direct Memory Access Controllers* on page 609.

DMA2P30 is attached to the 64-bit processor local bus. *Table 28-5* lists the DMA to PLB3 channel assignments.

Table 28-5. DMA to PLB3 Channel Assignments

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 0	Sub-Channel 0	UART 0 Receive	External Peripheral DMA 0
	Sub-Channel 1	UART 2 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 1	Sub-Channel 0	UART 0 Transmit	External Peripheral DMA 1
	Sub-Channel 1	UART 2 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

Table 28-5. DMA to PLB3 Channel Assignments (continued)

Channel	Sub-Channel	DMA Internal Source	DMA External Source
Channel 2	Sub-Channel 0	UART 1 Receive	External Peripheral DMA 2
	Sub-Channel 1	UART 3 Receive	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		
Channel 3	Sub-Channel 0	UART 1 Transmit	External Peripheral DMA 3
	Sub-Channel 1	UART 3 Transmit	
	Sub-Channel 2		
	Sub-Channel 3		
	Sub-Channel 4		
	Sub-Channel 5		
	Sub-Channel 6		
	Sub-Channel 7		

28.5.1 Transmitter DMA Mode

The four DMA channels on the PPC440EPx/GRx can be used to move data to and from the four UARTs. The UART0 and UART2 transmit channel is tied to DMA channel 1, the UART1 and UART3 transmit channel is tied to DMA channel 3. Use of the DMA with the UARTs is controlled by the UxDTE bits of the SDR0_UART0:3 registers.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when the FIFOs are disabled or the FIFOs are enabled and there are no characters in the TX FIFO or Transmit Holding Register (THR). Once activated, the DMA request goes inactive after the first character is loaded into the TX FIFO or THR.

When UARTx_FCR[DMS] is 1 the DMA request goes active, when FIFOs are enabled and there is at least one unfilled position in the TX FIFO. DMA request goes inactive when the TX FIFO is completely full. To operate in this mode the corresponding PPC440EPx/GRx DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA2P30_CRx to 1 configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 28-6 lists required register settings for UART0 transmit transfers. The UART0 transmit channel is tied to DMA channel 1. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-6. UART 0 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART0[U0DTE]=1	UART0 DMA transmit channel is enabled using DMA channel 1
SDR0_UART0[U0DC]	Set to 0 to not clear SDR0_UART0[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR1[TD]=0	DMA channel 1 transfer direction is from memory to peripheral.
DMA2P30_CR1[PL]=1	DMA channel 1 peripheral is on the OPB (UART0).
DMA2P30_CR1[PW]=000	Peripheral width is byte (8 bits).
DMA2P30_CR1[TM]=00	DMA Channel 1 is in peripheral mode.
DMA2P30_CR1[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR1[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR1[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC1[SCID] = 000	DMA Channel 1, sub-channel 0 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 1 for UART0 transmitter transfers, external DMA transfers cannot be performed on this channel.	

Table 28-7 lists required register settings for UART 1 transmit transfers. The UART1 transmit channel is tied to DMA channel 3. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-7. UART 1 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART1[U1DTE]=1	UART 1 DMA transmit channel is enabled using DMA channel 3.
SDR0_UART1[U1DC]	Set to 0 to not clear SDR0_UART1[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR3[TD]=0	DMA channel 3 transfer direction is from memory to peripheral.
DMA2P30_CR3[PL]=1	DMA channel 3 peripheral is on the OPB (UART0).
DMA2P30_CR3[PW]=000	Peripheral width is byte (8 bits).
DMA2P30_CR3[TM]=00	DMA channel 3 is in peripheral mode.
DMA2P30_CR3[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR3[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR3[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC3[SCID] = 000	DMA Channel 3 sub-channel 0 selected.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 3 for UART 1 transmitter transfers, external DMA transfers cannot be performed on this channel.	

Table 28-8 lists required register settings for UART2 transmit transfers. The UART2 transmit channel is tied to DMA channel 1. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-8. UART2 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART2[U2DTE]=1	UART2 DMA transmit channel is enabled using DMA channel 1
SDR0_UART2[U2DC]	Set to 0 to not clear SDR0_UART2[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR1[TD]=0	DMA channel 1 transfer direction is from memory to peripheral.
DMA2P30_CR1[PL]=1	DMA channel 1 peripheral is on the OPB (UART0).
DMA2P30_CR1[PW]=000	Peripheral width is byte (8 bits).
DMA2P30_CR1[TM]=00	DMA Channel 1 is in peripheral mode.
DMA2P30_CR1[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR1[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR1[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC1[SCID] = 001	DMA Channel 1 sub-channel 1 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 1 for UART0 transmitter transfers, external DMA transfers cannot be performed on this channel.	

Table 28-9 lists required register settings for UART3 transmit transfers. The UART3 transmit channel is tied to DMA channel 3. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-9. UART3 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART3[U3DTE]=1	UART 3 DMA transmit channel is enabled using DMA channel 3.
SDR0_UART3[U3DC]	Set to 0 to not clear SDR0_UART3[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR3[TD]=0	DMA channel 3 transfer direction is from memory to peripheral.
DMA2P30_CR3[PL]=1	DMA channel 3 peripheral is on the OPB (UART0).
DMA2P30_CR3[PW]=000	Peripheral width is byte (8 bits).
DMA2P30_CR3[TM]=00	DMA channel 3 is in peripheral mode.
DMA2P30_CR3[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA2P30_CR3[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR3[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC3[SCID] = 001	DMA Channel 3 sub-channel 1 selected.
UART3_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 3 for UART 3 transmitter transfers, external DMA transfers cannot be performed on this channel.	

28.5.2 Receiver DMA Mode

The four DMA channels on the PPC440EPx/GRx can be used to move data to and from the two UARTs. The UART0 and UART2 receive channel is tied to DMA channel 0 and the UART1 and UART3 receive channel is tied to DMA channel 2. Use of the DMA with the UARTs is controlled by the UxDRE bits of the SDR0_UART0 register.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when there is at least one character in the RX FIFO or Receive Buffer Register (RBR). Once activated, the DMA request goes inactive when there are no more characters in the FIFO or RBR.

When UARTx_FCR[DMS] is 1 the DMA request goes active when the FIFOs are enabled and the trigger level or the timeout has been reached. DMA request goes inactive when there are no more characters in the FIFO or RBR. To operate in this mode the corresponding PPC440EPx/GRx DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA2P30_CRx to 1configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 28-10 lists required register settings for UART0 receive transfers. The UART0 receive channel is tied to DMA channel 0. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-10. UART0 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART0[U0DRE]=1	UART0 DMA receiver channel is enabled using DMA channel 0.
SDR0_UART0[U0DC]	Set to 0 to not clear SDR0_UART0[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR0[TD]=1	DMA channel 0 transfer direction is from peripheral to memory.
DMA2P30_CR0[PL]=1	DMA channel 0 peripheral is on the OPB (UART0).
DMA2P30_CR0[PW]=000	Peripheral width is byte (8 bits).
DMA2P30_CR0[TM]=00	DMA channel 0 is in peripheral mode.
DMA2P30_CR0[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR0[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR0[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC0[SCID] = 000	DMA Channel 0 sub-channel 0 selected.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 0 for UART0 receiver transfers, external DMA transfers cannot be performed on this channel.	

Table 28-11 lists required register settings for UART1 receive transfers. The UART1 receive channel is tied to DMA channel 2. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-11. UART1 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART1[U1DRE]=1	UART1 DMA receiver channel is enabled using DMA channel 2.
SDR0_UART1[UIDC]	Set to 0 to not clear SDR0_UART1[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR2[TD]=1	DMA Channel 2 transfer direction is from peripheral to memory.
DMA2P30_CR2[PL]=1	DMA Channel 2 peripheral is on the OPB (UART0).
DMA2P30_CR2[PW]=00	Peripheral width is byte (8 bits).
DMA2P30_CR2[TM]=00	DMA Channel 2 is in peripheral mode.
DMA2P30_CR2[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR2[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR2[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC2[SCID] = 000	DMA Channel 2 sub-channel 0 selected.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 2 for UART1 receiver transfers, external DMA transfers cannot be performed on this channel.	

Table 28-12 lists required register settings for UART2 receive transfers. The UART2 receive channel is tied to DMA channel 0. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-12. UART2 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART2[U2DRE]=1	UART2 DMA receiver channel is enabled using DMA channel 0.
SDR0_UART2[U2DC]	Set to 0 to not clear SDR0_UART2[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR0[TD]=1	DMA channel 0 transfer direction is from peripheral to memory.
DMA2P30_CR0[PL]=1	DMA channel 0 peripheral is on the OPB (UART0).
DMA2P30_CR0[PW]=000	Peripheral width is byte (8 bits).
DMA2P30_CR0[TM]=00	DMA channel 0 is in peripheral mode.
DMA2P30_CR0[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR0[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR0[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC0[SCID] = 001	DMA Channel 0 sub-channel 1 selected
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 0 for UART2 receiver transfers, external DMA transfers cannot be performed on this channel.	

Table 28-13 lists required register settings for UART3 receive transfers. The UART3 receive channel is tied to DMA channel 2. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controllers* on page 609 for more information.

Table 28-13. UART3 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART3[U3DRE]=1	UART3 DMA receiver channel is enabled using DMA channel 2.
SDR0_UART3[U3DC]	Set to 0 to not clear SDR0_UART3[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA2P30_CR2[TD]=1	DMA Channel 2 transfer direction is from peripheral to memory.
DMA2P30_CR2[PL]=1	DMA Channel 2 peripheral is on the OPB (UART0).
DMA2P30_CR2[PW]=00	Peripheral width is byte (8 bits).
DMA2P30_CR2[TM]=00	DMA Channel 2 is in peripheral mode.
DMA2P30_CR2[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA2P30_CR2[PHC]=000	Peripheral hold cycles are 0.
DMA2P30_CR2[ETD]=1	EOT/TC is programmed as terminal count output.
DMA2P30_SC2[SCID] = 001	DMA Channel 2 sub-channel 1 selected
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.
Note: When using DMA channel 2 for UART1 receiver transfers, external DMA transfers cannot be performed on this channel.	

29. Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a full-duplex, synchronous, character-oriented (byte) port that enables the exchange of data with other serial devices. The SPI is master on the serial port supporting a 3-wire interface (receive, transmit, and clock). The SPI is also a slave device to the on-chip peripheral bus (OPB) for communication to the processor.

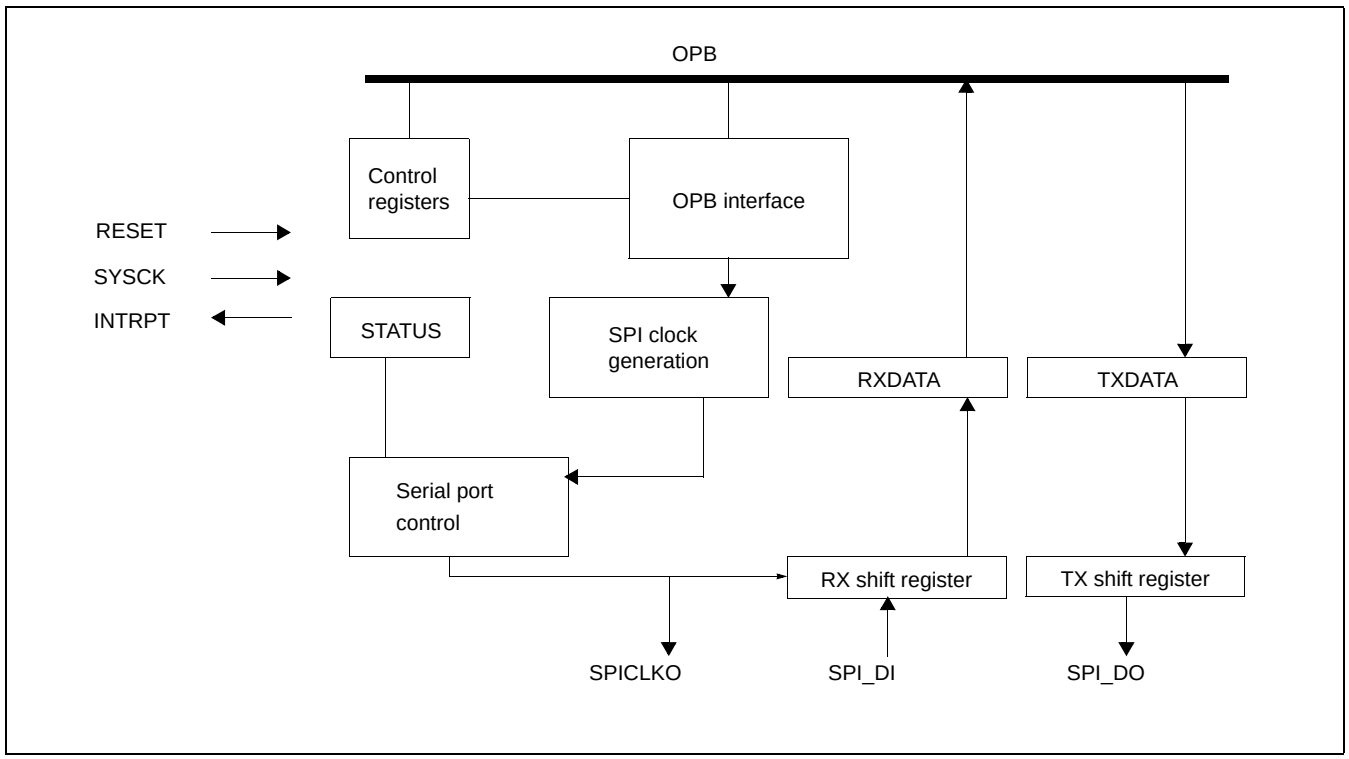
The Serial Peripheral Interface has the following features:

- Three-wire serial port interface
- Full-duplex synchronous operation
- SPI bus master
- OPB bus slave
- Programmable clock rate divider
- Clock inversion
- Optional clock phase
- Reverse data
- Local data loop back for test

29.1 Functional Description

Figure 29-1 shows the functional diagram of the port. The SPI accepts instructions from the OPB to read/write the control registers, or to read/write data from/to the data buffers, RxD and TxD. The SPI can then accept an instruction to initiate a transmit/receive cycle when SPI control register serial data transmit/receive bit is enabled (SPI0_CR[STP] = 1) (see *SPI Control Register (SPI0_CR)* on page 772). This activity is controlled by the OPB interface circuit.

Figure 29-1. SPI Functional Diagram



If a send/receive cycle is initiated by the OPB, then the serial port control circuit enables the SPI clock for a series of exactly eight clock pulses, shifting eight bits into and eight bits out of the RX and TX shift registers. When this is complete, the SPI clock is again disabled, and an interrupt is generated so the port can be serviced by the interrupt controller. The SPI is always in full duplex mode and always sends and receives one byte of data. The SPI clock is generated internally from the OPB clock (OPBClk) using a clock divider which can be programmed via the SPI0_CDM register (see *SPI Clock Divisor Modulus Register (SPI0_CDM)* on page 773).

The OPB interface uses 8-bit input and output data buses, referred to as OPB_DBUS[0:7] for input and SL_DBUS[0:7] for output. If an SPI register is addressed for write, then its bits are replaced by the respective bits in the OPB_DBUS. Unused control register bits are reserved and should always be written with 0.

If the SPI is addressed for a register read, then the register bits are placed onto the SL_DBUS and the unused or reserved bits are set to 0. For example, if the OPB reads the SPI0_MODE register, then the SL_DBUS is loaded with 00034567 where 34567 is the content of bits 3, 4, 5, 6, and 7 of the SPI0_MODE register. Reserved bits are set to 0 on a register read.

29.1.1 SPI Interrupt Operation

A single interrupt is generated when the send/receive cycle is complete (RxByteReady). The interrupt is cleared when the received data is read by the OPB. Therefore, even if the received data is not used by the processor, a read operation is still required to clear the interrupt.

29.1.2 Loopback

In loop back operation, the output data is inverted and received into the input receiver.

29.1.3 Typical Operation (Exchange of Data)

The SPI operates as a slave to the OPB. In a typical exchange of data, the processor initiates a data exchange with an external peripheral (an SPI slave). The processor activates the enable signal for that slave using one of the general purpose I/O pins.

To begin the exchange of data, the processor writes the data to be transmitted into the SPI0_TxD register, and then writes the start SPI0_SR[STR] register transmit bit to start the transmission of data. SPI0_SR[STR] is cleared by the SPI after transmission has begun on the serial port.

When the serial port controller recognizes the STR bit, it loads the TxData into the transmit shift register and generates eight clock pulses that simultaneously transmit and receive eight bits of data. The received data is then transferred to the RxData buffer, and an interrupt is issued to the interrupt controller. When the OPB reads the data from the SPI, the interrupt is cleared.

TxData is loaded and SPI0_SR[STR] is cleared when the send/receive cycle is started. Therefore, if SPI0_SR[STR] is written again before this register can be cleared, the new request is lost. If new transmit data is written to the TxData buffer before SPI0_SR[STR] is cleared (that is, old data not yet sent), then the old data in the TxData buffer is overwritten.

29.2 SPI Registers

SPI registers listed on *Table 29-1* are 8-bit wide memory mapped registers. Some registers have reserved bits which should be written as 0, and return 0 when read.

Table 29-1. SPI Registers

Mnemonic	R/W Mode	Address	Register	Page
SPI0_MODE	R/W	0x1 EF60 0900	SPI Mode Register	774
SPI0_RxD	R	0x1 EF60 0901	SPI Receive Data Register	772
SPI0_TxD	R/W	0x1 EF60 0902	SPI Transmit Data Register	772
SPI0_CR	R/W	0x1 EF60 0903	SPI Control Register	772
SPI0_SR	R	0x1 EF60 0904	SPI Status Register	773
SPI0_CDM	R/W	0x1 EF60 0906	SPI Clock Divisor Modulus Register	773

29.2.1 SPI Receive Data Register (SPI0_RxD)

The read-only SPI0_RxD register holds data received on the serial port.

Figure 29-2. SPI Receive Data Register (SPI0_RxD)

0	Rx0	Receive data 0	
1	Rx1	Receive data 1	
2	Rx2	Receive data 2	
3	Rx3	Receive data 3	
4	Rx4	Receive data 4	
5	Rx5	Receive data 5	
6	Rx6	Receive data 6	
7	Rx7	Receive data 7	

29.2.2 SPI Transmit Data Register (SPI0_TxD)

The read/write SPI0_TxD register holds data transmitted on the serial port.

Figure 29-3. SPI Transmit Data Register (SPI0_TxD)

0	Tx0	Transmit data 0	
1	Tx1	Transmit data 1	
2	Tx2	Transmit data 2	
3	Tx3	Transmit data 3	
4	Tx4	Transmit data 4	
5	Tx5	Transmit data 5	
6	Tx5	Transmit data 6	
7	Tx7	Transmit data 7	

29.2.3 SPI Control Register (SPI0_CR)

Figure 29-4 describes the read/write SPI0_CR register. Asserting SPI0_CR[STR] is a request that starts a new transmit/receive cycle on the serial port. When SPI0_CR[STR]=1, and the SPI is idle, a new byte of data is loaded into the output shift register and a new transmit/receive cycle begins. STR is cleared by the SPI when the transmit/receive begins. The delay between the setting of SPI0_CR[STR]=1 and the beginning of transmit is variable and depends on the SPI port frequency.

If the port is busy, SPI0_CR[STR] is a pending request and cleared when the next send/receive cycle begins. If SPI0_CR[STR] is asserted twice before it can be cleared, the request is lost.

Note: SPI0_CDM, SPI0_MODE, and SPI0_TxD registers should be written in order to configure the SPI before writing SPI0_CR.

Figure 29-4. SPI Control Register (SPI0_CR)			
0:6		Reserved	
7	STR	Serial data transmit/receive. Serial data transmit/receive disabled Serial data transmit/receive enabled	Reset to '0' after serial transmission has begun. Setting SPI0_CR[STR]=1 initiates transmit and receive of one character on the SPI.

29.2.4 SPI Clock Divisor Modulus Register (SPI0_CDM)

The SPI clock is generated from the OPB clock by a clock divider. The divide ratio is controlled by the value of the SPI0_CDM register, which is initially reset to 00000000. SPI0_CDM represents a programmable binary number (with bit 0 as the most significant bit (MSb)) that is used for the initial value of a count down register. When the count equals zero, the counter is again loaded with the value of SPI0_CDM. This counter is used to generate the output SPI clock (SCPClkOut). The output clock frequency is given by the ratio:

SCPClkOut = OPBCLK/(4(CDM+1)) where CDM = 0–255.

Therefore, the minimum divide ratio is 4 when CDM = 0, and the maximum is 1024 when CDM = 255.

Figure 29-5. SPI Clock Divisor Modulus Register (SPI0_CDM)			
0	CDM0	Clock divisor modulus 0	
1	CDM1	Clock divisor modulus 1	
2	CDM2	Clock divisor modulus 2	
3	CDM3	Clock divisor modulus 3	
4	CDM4	Clock divisor modulus 4	
5	CDM5	Clock divisor modulus 5	
6	CDM6	Clock divisor modulus 6	
7	CDM7	Clock divisor modulus 7	

29.2.5 SPI Status Register (SPI0_SR)

Figure 29-6 describes the SPI0_SR register bits. After a transmit/receive cycle is complete, the received data is loaded into the RxData buffer. The SPI0_SR[RBR] is asserted and this creates an interrupt. If RxData already contained previously received data, then that data is overwritten. SPI0_SR[RBR] also confirms that the transmit byte has been sent. SPI0_SR[BSY] is asserted one cycle after the start of send/receive and de-asserted one cycle after the send/receive is complete.

Figure 29-6. SPI Status Register (SPI0_SR)			
0:5		Reserved	
6	BSY	Busy Serial port is idle. Serial transmit/receive is active.	
7	RBR	RxByteReady RxD buffer is empty. Serial data receive is complete and RxD is available.	

29.2.6 SPI Mode Register (SPI0_MODE)

SPI is configured via SPI0_MODE register as described in *Figure 29-7* (SPI0_MODE[SPE]). If the other bits of SPI0_MODE and SPI0_CDM are changed while the serial port is active, the behavior will not be predictable. When the SPI0_MODE register is at its reset value (00000000), the output clock, SCPClkOut, is forced to 0.

Figure 29-7. SPI Mode Register (SPI0_MODE)			
0:2		Reserved	
3	SCP	Serial Clock Phase Data is latched on trailing edge of clock Data is latched on leading edge of clock	
4	SPE	Serial Port Enabled Serial port disabled Serial port enabled	SPI0_MODE[SPE] is used to enable the SPI port operation. When SPI0_MODE[SPE]=0 the serial port control circuits are put into a known reset state. When SPI0_MODE[SPE]=1, the port is enabled and ready to initiate a send/receive cycle. This bit must be asserted before transmission is started (writing to the SPI0_CR register). If SPI0_MODE[SPE]=0 while the port is transmitting, the current transmit cycle is completed before SPI0_MODE[SPE]=0 is recognized.
5	RD	Reverse Data Data Bit 0 is transmitted first. Data Bit 7 is transmitted first.	
6	CI	Clock Invert Idle clock = '0', Active clock = '1'. Idle clock = '1', Active clock = '0'.	
7	IL	Internal Loopback Loopback is disabled. Loopback is enabled.	When SPI0_MODE[IL]=1 the serial output data is inverted and connected to the receiver input.

30. IIC Bus Interface

The PPC440EPx/GRx provides two inter-integrated circuit (IIC) bus interfaces, IIC0 and IIC1. These interfaces comply with the specifications contained in the Phillips ® Semiconductors document *The I^2C Bus and How to Use It* (including specifications—1995 update).

Each IIC bus has a two wire, bi-directional, open-drain, low-speed serial interface. The serial clock (IIC0SClk and IIC1SClk) and serial data (IIC0SDA and IIC1SDA) lines are bidirectional, to support multiple bus masters and to mix high- and low-speed devices on the same bus.

Each IIC interface (referred to as IIC to distinguish it from the Phillips I 2 C bus) supports the following standard and enhanced features:

- 100 kHz and 400 kHz operation
- 8-bit data transfers
- 7 bit and 10 bit addressing
- Slave transmitter and receiver
- Master transmitter and receiver
- Multiple bus masters

30.1 Addressing

The IIC interfaces support 7-bit and 10-bit addressing for master and slave transfers. Addressing is described in detail in *IICx Low Master Address Register (IICx_LMADR)* on page 779, *IICx High Master Address Register (IICx_HMADR)* on page 780, *IICx Low Slave Address Register (IICx_LSADR)* on page 785, and *IICx High Slave Address Register (IICx_HSADR)* on page 786.

Descriptions of addressing modes and address formats follow.

30.1.1 Addressing Modes

For master transfers, the address mode (AMD) field of the IIC Control register (IICx_CNTL) controls whether 7-bit or 10-bit addresses are used. If IICx_CNTL[AMD] = 0, addresses contain 7 bits; if IICx_CNTL[AMD] = 1, addresses contain 10 bits.

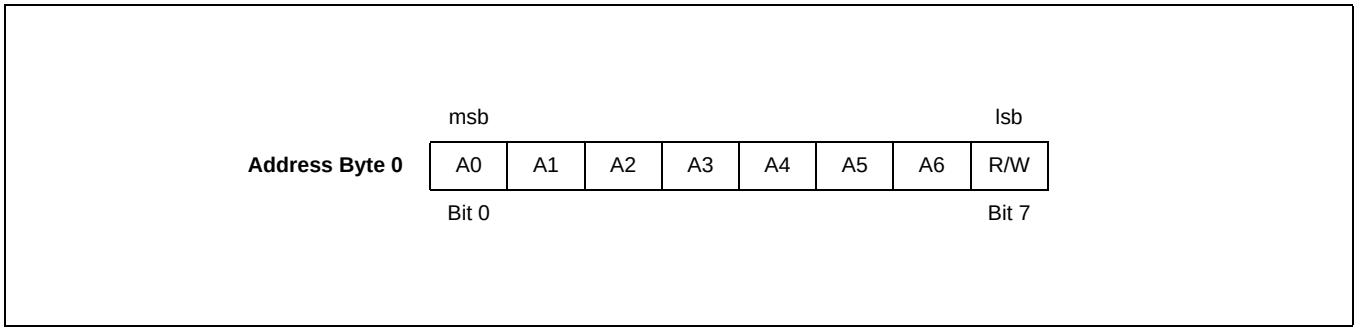
For slave transfers, the contents of the IICx High Slave Address register (IICx_HSADR) determines whether 7-bit or 10-bit addressing is used. If IICx_HSADR = 0b00000000, 7-bit addressing is used. If 10-bit addressing is to be used for slave transfers, IICx_HSADR = 0b11110*yy*x, where *yy* contains the high-order bits of the 10-bit address, and x is a don't care.

Programming Note: For slave transfers, IICx_CNTL[AMD] does not control addressing mode.

30.1.2 Seven-Bit Addresses

Figure 30-1 illustrates a 7-bit address. For master transfers, the address bits 0 through 6 (A0:A6) are read from IICx_LMADR. For slave transfers, A0:A6 are read from IICx_LSADR. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

Figure 30-1. 7-Bit Addressing



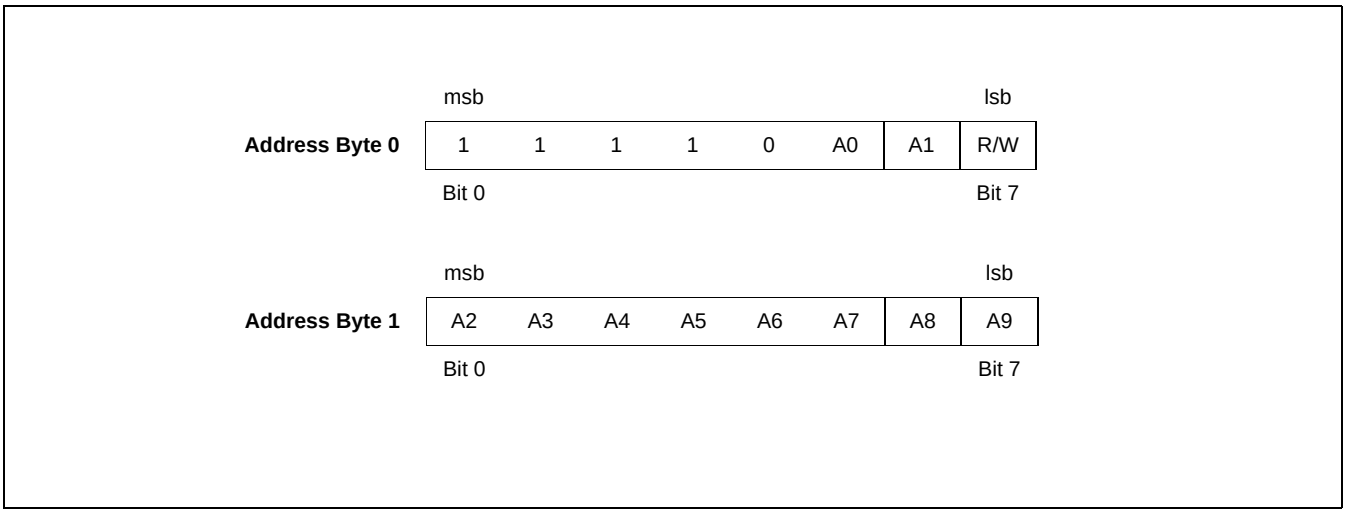
30.1.3 Ten-Bit Addresses

Figure 30-2 illustrates a 10-bit address. A0:A1 of address byte 0 are read from IICx_HMADR[A6:A7] (for master transfers) or IICx_HSADR[A6:A7] (for slave transfers). These are the two highest-order address bits transmitted on the IIC bus. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

For 10-bit addressing for master or slave transfers, respectively, IICx_HMADR[A0:A4] and IICx_HSADR [A0:A4] must contain 0b11110.

The low-order byte of the 10-bit address, contained in A0:A7 of address byte 1, are read from IICx_LMADR or IICx_LSADR for master or slave transfers, respectively.

Figure 30-2. 10-Bit Addressing



30.2 IIC Registers

Table 30-1 lists the IICx registers. Descriptions of the registers, in the listed order, follow in *IIC Register Descriptions* on page 778.

Table 30-1. IIC Registers

Register	Description	Address	Access	Effect of Reset	Bits	Page
IIC0_MDBUF	IIC0 Master Data Buffer	0x1 EF60 0700	R/W	Cleared	8,16	778
IIC0_SDBUF	IIC0x Slave Data Buffer	0x1 EF60 0702	R/W	Cleared	8,16	778
IIC0_LMADR	IIC0 Low Master Address	0x1 EF60 0704	R/W	No	8	779
IIC0_HMADR	IIC0 High Master Address	0x1 EF60 0705	R/W	No	8	780
IIC0_CNTL	IIC0 Control	0x1 EF60 0706	R/W	Cleared	8	780
IIC0_MDCNTL	IIC0 Mode Control	0x1 EF60 0707	R/W	Cleared	8	781
IIC0_STS	IIC0 Status	0x1 EF60 0708	R/W	Cleared	8	782
IIC0_EXTSTS	IIC0 Extended Status	0x1 EF60 0709	R/W	Cleared	8	784
IIC0_LSADR	IIC0 Low Slave Address	0x1 EF60 070A	R/W	No	8	785
IIC0_HSADR	IIC0 High Slave Address	0x1 EF60 070B	R/W	No	8	786
IIC0_CLKDIV	IIC0 Clock Divide	0x1 EF60 070C	R/W	Cleared	8	786
IIC0_INTRMSK	IIC0 Interrupt Mask	0x1 EF60 070D	R/W	Cleared	8	787
IIC0_XFRCNT	IIC0 Transfer Count	0x1 EF60 070E	R/W	Cleared	8	788
IIC0_XTCNTLSS	IIC0 Extended Control and Slave Status	0x1 EF60 070F	R/W	Cleared	8	789
IIC0_DIRECTCNTL	IIC0 Direct Control	0x1 EF60 0710	R/W	0x0F	4	790
IIC0_INTR	IIC0 Interrupt	0x1 EF60 0711	R	Cleared	2	791
IIC1_MDBUF	IIC1 Master Data Buffer	0x1 EF60 0800	R/W	Cleared	8,16	778
IIC1_SDBUF	IIC1 Slave Data Buffer	0x1 EF60 0802	R/W	Cleared	8,16	778
IIC1_LMADR	IIC1 Low Master Address	0x1 EF60 0804	R/W	No	8	779
IIC1_HMADR	IIC1 High Master Address	0x1 EF60 0805	R/W	No	8	780
IIC1_CNTL	IIC1 Control	0x1 EF60 0806	R/W	Cleared	8	780
IIC1_MDCNTL	IIC1 Mode Control	0x1 EF60 0807	R/W	Cleared	8	781
IIC1_STS	IIC1 Status	0x1 EF60 0808	R/W	Cleared	8	782
IIC1_EXTSTS	IIC1 Extended Status	0x1 EF60 0809	R/W	Cleared	8	784
IIC1_LSADR	IIC1 Low Slave Address	0x1 EF60 080A	R/W	No	8	785
IIC1_HSADR	IIC1 High Slave Address	0x1 EF60 080B	R/W	No	8	786
IIC1_CLKDIV	IIC1 Clock Divide	0x1 EF60 080C	R/W	Cleared	8	786
IIC1_INTRMSK	IIC1 Interrupt Mask	0x1 EF60 080D	R/W	Cleared	8	787
IIC1_XFRCNT	IIC1 Transfer Count	0x1 EF60 080E	R/W	Cleared	8	788
IIC1_XTCNTLSS	IIC1 Extended Control and Slave Status	0x1 EF60 080F	R/W	Cleared	8	789

Table 30-1. IIC Registers (continued)

Register	Description	Address	Access	Effect of Reset	Bits	Page
IIC1_DIRECTCNTL	IIC1 Direct Control	0x1 EF60 0810	R/W	0x0F	4	790
IIC1_INTR	IIC0 Interrupt	0x1 EF60 0811	R	Cleared	2	791

30.3 IIC Register Descriptions

The following sections contains the bit definitions for the various registers in the IIC interface.

30.3.1 IICx Master Data Buffer (IICx_MDBUF)

The IICx Master Data Buffer (IICx_MDBUF) is a 1-byte × 4-byte first-in/first-out (FIFO) buffer. Data placed in IICx_MDBUF is written onto the IIC bus when performing a master write operation. Data received from the IIC bus is read from IICx_MDBUF when performing a master read operation.

Figure 30-3. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit	
1		Data bit	
2		Data bit	
3		Data bit	
4		Data bit	
5		Data bit	
6		Data bit	
7		Data bit	

Half word reads from and writes to the IIC bus assume the MSB comes first. For half word writes, the first byte written to the IICx_MDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For half word reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_MDBUF while the FIFO is empty, obsolete data is read. Check the master buffer status, IICx_STS[MDBS], before reading IICx_MDBUF.

If a byte is written to IICx_MDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the master buffer full status, IICx_STS[MDBF], before writing IICx_MDBUF.

IICx_MDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FMDB] = 1.

30.3.2 IICx Slave Data Buffer (IICx_SDBUF)

The IICx Slave Data Buffer (IICx_SDBUF) is a 1-byte 4-byte first-in/first-out (FIFO) buffer. The data contained in IICx_SDBUF is either received from the IIC bus when the IIC interface is addressed as a slave during a write, or is written on the IIC bus when the IIC interface is addressed as a slave during a read operation.

Figure 30-4. IICx Slave Data Buffer (IICx_SDBUF)			
0		Data bit	
1		Data bit	
2		Data bit	
3		Data bit	
4		Data bit	
5		Data bit	
6		Data bit	
7		Data bit	

Half word reads from and writes to the IIC bus assume the MSB comes first. For half word writes, the first byte written to the IICx_SDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For half word reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_SDBUF while the FIFO is empty, obsolete data is read. Check the slave buffer status, IICx_XTCNTLSS[SDBD], before reading IICx_SDBUF.

If a byte is written to IICx_SDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the slave buffer full status, IICx_XTCNTLSS[SDBF], before writing IICx_SDBUF.

IICx_SDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FSDB] = 1.

30.3.3 IICx Low Master Address Register (IICx_LMADR)

The IICx Low Master Address (IICx_LMADR) and IICx High Master Address Register (IICx_HMADR) form addresses that the IIC interface transmits on the IIC bus.

When in 7-bit address mode, IICx_CNTL[AMD] = 0, IICx_LMADR[A0:A6] is the address transmitted on the IIC bus; IICx_LMADR[A7] is a don't care. When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_LMADR[A0:A7] is the second byte of the address transmitted on the IIC bus.

Figure 30-5. IICx Low Master Address Register (IICx_LMADR)			
0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

30.3.4 IICx High Master Address Register (IICx_HMADR)

IICx High Master Address Register (IICx_HMADR) provides the upper address bits in 10-bit addressing mode.

When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_HMADR must be programmed to 0b1111 0yyz, where yy are the high-order bits of a 10-bit address and z is a don't care.

IICx_HMADR[A5:A6] are the two most significant bits of the 10-bit address. IICx_HMADR[A7] is a don't care.

Figure 30-6. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

30.3.5 IICx Control Register (IICx_CNTL)

The IICx Control Register (IICx_CNTL) starts and stops IIC interface master transfers on the IIC bus. When a transfer begins, the IIC interface uses the values in IICx_CNTL to determine the type and size of the transfer.

> **Programming Note:** IICx_CNTL *must* be the last register programmed. Whenever IICx_CNTL[PT] = 1, the IIC interface attempts to perform the requested transfer using values set in other registers.

During and after transfers, the IICx_STS and IICx_EXTSTS registers can be read to determine the state of the IIC interface and the IIC bus.

Only IICx_CNTL[PT] is cleared when a requested master transfer is complete; the remaining bits are not affected.

Figure 30-7. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	

5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

Table 30-2 summarizes IIC interface operation for settings of IICx_CNTL[HMT, RPST, CHT, PT]. x represents a don't care in the RPST, CHT and PT columns.

Table 30-2. IIC Response to IICx_CNTL Field Settings

IICx_CNTL Fields				Resulting Action on IIC Bus and Inside IIC Interface
HMT	RPST	CHT	PT	
0	x	x	0	No action taken
0	0	1	1	Start, Transfer, ACK on last byte, Pause
0	0	0	1	Start, Transfer, NACK on last byte, Stop
1	x	x	x	NACK on current byte, Stop
0	1	x	1	Start, Transfer, NACK on last byte, Wait

Settings of IICx_CNTL[HMT, RPST, CHT, PT] result in the following actions.

Start IIC Start condition generated, if the IIC interface was stopped or waiting.

Stop IIC Stop condition generated; IIC interface enters the Stop condition.

ACK IIC Acknowledge condition generated.

NACK IIC Not Acknowledge condition generated, if performing a read.

Transfer Requested bytes are transferred.

Pause IIC interface enters the Pause state.

Wait IIC interface enters the Wait state.

IICx_CNTL[HMT] overrides IICx_CNTL[RPST, CHT].

IICx_CNTL[RPST, PT] overrides IICx_CNTL[CHT].

30.3.6 IICx Mode Control Register (IICx_MDCNTL)

The IICx Mode Control Register (IICx_MDCNTL) sets the major modes of operation on the IIC bus. In addition, IICx_MDCNTL can force the data buffers into the empty state.

In typical applications, IICx_MDCNTL is configured once, during software initialization. Applications providing complex error handling may reconfigure this register more often.

> **Programming Note:** IICx_CLKDIV must be initialized before IICx_MDCNTL. IICx_LSADR and IICx_HSADR should also be configured before IICx_MDCNTL.

Note that the IIC hardware does not implement time-out functions on the IIC bus. Such functions must be implemented, in software, by setting IICx_CNTL[HMT] = 1, or setting IICx_XTCNTLSS[SRST] = 1.

Regarding IICx_MDCNTL[HSCL], a "slave not ready" condition occurs during a slave receive operation, if there is no space in the slave data buffer at the start of a write operation, or if the slave data buffer fills during the write. In a slave transmit operation, a slave not ready condition occurs if there is no data in the slave data buffer at the start of a read operation, or if the slave data buffer becomes empty during the read.

Using IICx_MDCNTL[HSCL] to handle slave not ready conditions can affect system performance. A slave holding the IICSCL signal low guarantees data delivery to or from the requesting master, but prevents other masters from performing transfers over the IIC bus. There is no general rule for handling slave not ready conditions; each system has its own requirements.

Figure 30-8. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICSCL signal low until slave is ready.	This field is used only when in slave mode.

30.3.7 IICx Status Register (IICx_STS)

The IICx Status register (IICx_STS) contains the state of the IIC interface and the status of any previously requested master transfers.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Programming Note: IICx_STS should be the first register read by an interrupt or error handler routine. IICx_STS can also be read in a polling loop if the IIC interrupts are not used.

All IICx_STS bit fields except for IICx_STS[SSS] must be cleared before requesting another master transfer. IICx_STS[SSS] is not affected by master transactions.

Figure 30-9. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRC, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (SDR0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the SDR0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

The Error and Pending Transfer, IICx_STS[ERR, PT], bit fields indicate the success or failure of the requested transfer. *Table 30-3* lists the transfer status for all combinations of the IICx_STS[ERR,PT] bit fields.

Table 30-3. IICx_STS[ERR, PT] Decoding

ERR	PT	Status
0	0	Requested transfer completed without errors
0	1	Requested transfer is in progress; no errors were detected
1	0	Requested transfer is complete, but not all data was transferred
1	1	Requested transfer is in progress; but an error was detected

Programming Note: No action regarding a master transfer should be taken unless all pending transfers have completed, IICx_STS[PT] = 0.

If an error requires the IIC interface to send a Stop, the Stop Complete bit field is set, IICx_STS[SCMP] = 1. Note that slave operations should be serviced regardless of the state of a requested master transfer.

IIC1_MDCNTL[EUBS] must be set after a reset before IIC interface can be placed in sleep mode. The IIC interface is placed in sleep mode by setting the CPM0_ER[IICx] via software. Awaking the IIC interface is possible directly through software by clearing the CPM0_ER[IICx] or indirectly by detecting a Start condition on the IIC bus. When a Start condition is detected, the IIC interface is awakened, clearing the CPM0_ER[IICx] and the IICx_STS[SLPR].

The IICx_STS[MDBS, MDBF] contain the current status of the Master Data Buffer, IICx_MDBUF. When the IICx_MDBUF contains data, IICx_STS[MDBS] is set. When the IICx_MDBUF is full, IICx_STS[MDBF] is set.

The state of the IICx_MDBUF is not instantly recorded by the IICx_STS[MDBS, MDBF]. The delay depends on the size of the buffer access. For half word accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

30.3.8 IICx Extended Status Register (IICx_EXTSTS)

The IICx Extended Status register (IICx_EXTSTS) reports additional IIC status.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Figure 30-10. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. • Writing 1 to IICx_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	Bus Control State 000 Unused; if this value is read an error occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read an error occurred.	Read-only.

4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.
7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

IICx_EXTSTS[IRQP, IRQD] and IICx_STS[IRQA] provide a FIFO for storing interrupts. A new interrupt is considered pending, and remains pending while an on-deck interrupt is present. Once the on-deck interrupt becomes active, the pending interrupt moves on-deck, and remains on-deck until there is no active interrupt. When the active interrupt is cleared, the on-deck (initially pending) interrupt becomes active.

Programming Note: An active interrupt remains active until software clears it.

IICx_EXTSTS[BCS] indicates the state of the IIC interface. The field is read-only.

IICx_EXTSTS[LA, ICT, XFRA] are cleared when IICx_EXTSTS[XFRA] = 1.

Writing 1 to IICx_EXTSTS[IRQP, IRQD, LA, ICT, XFRA] clears these fields.

30.3.9 IICx Low Slave Address Register (IICx_LSADR)

The IICx Low Slave Address Register (IICx_LSADR) and IICx_ High Slave Address Register (IICx_HSADR) program the slave address of the IIC interface. IICx_HSADR is used only for 10-bit addressing, and is not programmed in 7-bit addressing mode.

When 7-bit addressing is used, IICx_LSADR is written with the slave address; IICx_HSADR must be written with zeros. For 7-bit addressing, IICx_LSADR[A0:A6] contain the address transmitted on the IIC bus; IICx_LSADR[A7] is a don't care.

When 10-bit addressing is used, IICx_LSADR[A0:A7] contain the second address byte transmitted on the IIC bus.

Figure 30-11. IICx Low Slave Address Register (IICx_LSADR)			
0	A0	Address bit 0	
1	A1	Address bit 1	
2	A2	Address bit 2	
3	A3	Address bit 3	
4	A4	Address bit 4	
5	A5	Address bit 5	
6	A6	Address bit 6	LSb for 7-bit addresses
7	A7	Address bit 7	LSb for 10-bit addresses; don't care for 7-bit addresses

30.3.10 IICx High Slave Address Register (IICx_HSADR)

For 7-bit addressing, set IICx High Slave Address Register (IICx_HSADR) to 0.

To enable 10-bit slave addressing, IICx_HSADR must be programmed to 0b1111 0*yyx*, where *yy* are the high-order bits of a 10-bit address and *x* is a don't care.

Programming Note: IICx_HSADR is used only for 10-bit addressing, and should be set to 0 for 7-bit addressing mode.

Thus, in 10-bit address mode, IICx_HSADR[A6:A7] contain the two highest -order bits of the 10-bit address; IICx_HSADR[A7] is a don't care. IICx_LSADR contains the low-order byte of the 10-bit address.

Figure 30-12. IICx High Slave Address Register (IICx_HSADR)			
0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

Thus, in 10-bit address mode, bits 0:6 are used to decode the first address byte that was transmitted on the IIC bus, and bit 7 is in a don't care.

30.3.11 IICx Clock Divide Register (IICx_CLKDIV)

The IICx Clock Divide Register (IICx_CLKDIV) establishes a reference between the OPB clock and the IIC bus serial clock.

Programming Note: IICx_CLKDIV must be initialized before IICx_MDCTRL. Until IICx_CLKDIV is initialized, all IIC bus activity is ignored.

Figure 30-13. IICx Clock Divide Register (IICx_CLKDIV)			
0	DIV0	Divisor bit 0	
1	DIV1	Divisor bit 1	
2	DIV2	Divisor bit 2	
3	DIV3	Divisor bit 3	
4	DIV4	Divisor bit 4	
5	DIV5	Divisor bit 5	
6	DIV6	Divisor bit 6	
7	DIV7	Divisor bit 7	

IICx_CLKDIV divides PPC440EPx/GRx's on-chip peripheral bus (OPB) clock to form the base clock for the IIC bus.

Table 30-4 lists the divisor values for several OPB frequency ranges. These divisor values apply for standard and fast mode. Select the divisor value by matching the OPB clock frequency to the corresponding frequency range in *Table 30-4*. For example, if the OPB clock frequency is 50MHz, select a divisor value of 0x4.

Table 30-4. IICx Clock Divide Programming

OPB Frequency Range (MHz)	Divisor Value
20	0x1
$20 < f \leq 30$	0x2
$30 < f \leq 40$	0x3
$40 < f \leq 50$	0x4
$50 < f \leq 60$	0x5
$60 < f \leq 70$	0x6
$70 < f \leq 80$	0x7
$80 < f \leq 90$	0x8
$90 < f \leq 100$	0x9
$100 < f \leq 110$	0xA
$110 < f \leq 120$	0xB
$120 < f \leq 130$	0xC
$130 < f \leq 140$	0xD
$140 < f \leq 150$	0xE

30.3.12 IICx Interrupt Mask Register (IICx_INTRMSK)

The IICx Interrupt Mask Register (IICx_INTRMSK) specifies which conditions can generate an IIC interrupt when the IIC interrupt is enabled, IICx_MDCNTL[EINT]=1.

Figure 30-14. IICx Interrupt Mask Register (IICx_INTRMSK)			
0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SRC] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. **Note:** IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. **Note:** IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

30.3.13 IICx Transfer Count Register (IICx_XFRCNT)

The IICx Transfer Count Register (IICx_XFRCNT) reports the number of bytes transferred on the IIC bus during a master or a slave operation.

Figure 30-15. IICx Transfer Count Register (IICx_XFRCNT)			
0		Reserved	
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved	
4		Reserved	

5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved	

IICx_XFRCNT[MTC] is cleared when there is a pending transfer, IICx_CNTL[PT] = 1.

IICx_XFRCNT[STC] is cleared when:

- A slave operation starts on the IIC bus
- Software indicates the slave does not need service by clearing IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SWS].

30.3.14 IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

The IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) provides additional control of IIC interface functions and reports the status of slave operations.

Figure 30-16. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IICx_SDBUF.
1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set. 2. If ICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data. 3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IICSCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data. 4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.
2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	

3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IICx_SDBUF is full during a slave write.	1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IICSCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.
4	SDBD	Slave Data Buffer Has Data 0 IICx_SDBUF is empty 1 IICx_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

Writing a 1 to IICx_XTCNTLSS[SRC, SRS, SWC, SWS] clears these fields.

The IICx_XTCNTLSS[SBSS, SDBF] contain the current status of the Slave Data Buffer, IICx_SDBUF. When the IICx_SDBUF contains data, IICx_XTCNTLSS[SDBD] is set. When the IICx_SDBF is full, IICx_XTCNTLSS[SDBF] is set.

The state of the IICx_SDBUF is not instantly recorded by the IICx_XTCNTL[SDBD, SDBF]. The delay depends on the size of the buffer access. For half-word accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

If any of the following fields: IICx_XTCNTLSS[SRC, SRS, SWC, SWS] = 1 and IICx_MDCNTL[HSCL] = 0; no new slave operations will be accepted over the IIC bus. A NACK is issued until IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], and IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], are cleared.

Soft reset, IICx_XTCNTLSS[SRST], provides a last means of recovery from IIC interface or IIC bus failure. Once enabled, soft reset completely resets the IIC interface. All IIC registers are affected. All transmissions from the IIC interface are terminated. Enabling soft reset during an IIC transmission may improperly terminate the transmission and hang the IIC bus.

30.3.15 IICx Direct Control Register (IICx_DIRECTCNTL)

The IICx Direct Control Register (IICx_DIRECTCNTL), which controls and monitors the IIC serial clock (IICSCL) and serial data (IICSDA) signal, is used for error recovery when a malfunction is detected on the IIC interface.

Figure 30-17. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved	
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1	

5	SCLC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSCL	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1	Read-only

IICx_DIRECTCNTL[SDAC, SCLC] can be written to control the IICSDA and IICSCL signals. When controlling the IICSDA and IICSCL signals directly, the IIC controller must be placed in the reset state, IICx_XTCNTL[SRST] = 1.

IICx_DIRECTCNTL[MSDA, MSCL] are used to verify that IICx_DIRECTCNTL[SDAC, SCLC] were written successfully, and that the IICSCL signals can be controlled. If IICx_DIRECTCNTL[MSDA, MSCL] do not correspond to IICx_DIRECTCNTL[SDAC, SCLC], respectively, toggle IICSCL repeatedly to regain control.

IICx_DIRECTCNTL[SDAC, SCLC, MDSA, MSCL] = 1 after a chip or system reset. A Soft Reset, IICx_XTCNTLSS[SRST] = 1, does not affect the state of IICx_DIRECRCNTL.

30.3.16 IICx Interrupt Register (IICx_INTR)

This register is used for diagnostic purposes, providing information on queued or pending interrupts. This register is not visible on the OPB if Enable_32byte_window is false.

Bit 1 is cleared when the on-deck interrupt is cleared and the pending IRQ has become the on-deck IRQ. Bit 0 is cleared when the active interrupt is cleared and the on-deck IRQ has become the active IRQ. Both bits are cleared by a hard or a soft reset.

The pending and on-deck interrupts, along with the active interrupt in the status register, form a miniature FIFO for storing the interrupts. A new interrupt is first set into the pending state. It will stay pending as long as an on-deck interrupt is present. Once the on-deck interrupt is cleared, or if none was present at the time the new interrupt occurred, the pending interrupt is moved into the on-deck state. An on-deck interrupt remains in the on-deck state as long as an active interrupt is present. Once the active interrupt is cleared, or if none was present at the time the pending interrupt went to the on-deck state, the on-deck interrupt is moved into the active state. Note that an active interrupt remains in the active state until it is cleared by the program.

Figure 30-18. IICx Interrupt Register (IICx_INTR)

0	IRQD	IRQ On-Deck Set when an IRQ is still active and another interrupting condition was generated.	This bit might also be momentarily set while a new interrupting condition moves from the on-deck to the active state. When IRQ active is set to 0, an on-deck IRQ will cause IRQ active to be set 1, and IRQ on-deck will be cleared. If there is a pending IRQ, then IRQ on-deck will be set to 1 on the next system clock. The on-deck interrupt can be cleared by writing 1 to this bit. If interrupts are disabled, then bit 2 in the Mode Control Register is 0, and this bit will not be set.

1	IRQP	IRQ Pending Set when an IRQ is still active, an IRQ is on-deck, and another interrupting condition was generated.	This bit might also be momentarily set while a new interrupting condition moves from the pending to the on deck state. When IRQ on-deck is set to 0, a pending IRQ will cause IRQ on-deck to be set to 1, and IRQ pending will be cleared. The pending interrupt can be cleared by writing 1 to this bit. If interrupts are disabled, then bit 2 in the Mode Control Register is 0, and this bit will not be set.
2:7		Reserved	

30.4 Interrupt Handling

Service request on the IIC interface can be monitored by polling the IICx_STS[SSS, PT] or by using the IIC interrupt to the UIC. When the IICx_MDCNTL[EINT] is enabled, the IIC interface generates an interrupt to the UIC if an unmasked IIC interrupt condition occurs. The interrupt is recorded by the UIC if UIC0_ER[IICx] is enabled. The conditions that generate an IIC interrupt to the UIC are determined by the interrupt mask settings in IICx_INTMSK.

The IIC interface can queue up to three interrupts since there is one interrupt for master operations and two for slave operations. The current interrupt is referred to as the *active* interrupt. The first interrupt in the queue is the on-deck interrupt; the second queued interrupt is called the pending interrupt. The queue holds multiple interrupts until the active interrupt is cleared by writing a 1 to IICx_STS[IRQA]. When an active interrupt is cleared, the on-deck interrupt becomes the active interrupt and the pending interrupt becomes the on-deck interrupt.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

Under certain conditions, the IIC interface merges slave read (write) needs service and slave read (write) complete interrupts into one interrupt. If a slave read (write) needs service interrupt is active, or queued, and a slave read (write) complete interrupt occurs, and IICx_XTCNTLSS has not yet been read, the two interrupts are merged into a single interrupt. This merge function is performed in the IIC interface logic, and is not under software control.

30.5 General Considerations

1. After a reset, the IIC interface enters the unknown IIC bus state. This state is exited when either activity is seen on the bus or when the exit unknown IIC bus state bit (in the mode control register), is set to a 1. If the IIC interface is being used in a single master system as the master, then the exit unknown IIC bus state bit must be used to force the logic out of the unknown state.
2. Once a byte is written into either the master or slave buffer, a total of four OPB clock periods must occur before the data can be read. Flushing the master or slave buffer also requires four OPB clock periods to complete.
3. IICx_DIRECTCNTL [MSDA,MSC]are used to verify that IICx_DIRECTCNTL [SDAC,SCC] were written successfully,and that the IICSCL signals can be controlled.If IICx_DIRECTCNTL [MSDA, MSC] do not correspond to IICx__DIRECTCNTL [SDAC,SCC],respectively, toggle IICSCL repeatedly to regain control.

4. The master and slave buffers are 4 × 1 byte-wide FIFOs. Exercise care when using master and slave buffers. As an example, consider the case where one byte of data is written on the IIC bus. The data is first written into the master or slave buffer. After four OPB clock cycles the data is placed on the IIC bus. There is no way to verify data in the buffer without disturbing the IIC transaction. The act of verification requires removing the data. If the data is removed, invalid data is placed on the IIC bus.
5. Use care when monitoring the IICx_XCNTLSS[SDBD] or to IICx_STS[MDBS] to determine when data is present. These bits are set to 1 when the buffer contains data in any stage. Consider the case where the master buffer is empty prior to being loaded with a byte received over the IIC bus. The byte enters the fourth stage of the buffer and the IICx_STS[MDBS] is set to 1. Stages 1, 2, and 3 do not contain data. Therefore, the data is not available for four OPB clock cycles. Any attempt to prematurely read the data yields invalid data.
6. When responding to a slave needs service request, manage the data first. Read data out of the slave buffer, IICx_SDB, for slave reads or write data into the IICx_SDB for slave writes. Next clear the slave needs service request. For reads, clear IICx_XTCNTLSS[SRS]. For writes, clear IICx_XTCNTLSS[SWS]. Last, clear the active interrupt, IICx_STS[IRQA] =0.
7. There is no timeout function implemented in the IIC interface. If this type of error recovery function is needed, it must be implemented in software.
8. Avoid the situations listed in Section 7.2 of the *Phillips Semiconductors I^{2}C Specification,* dated 1995. For your convenience, the section is summarized as follows:

   If multiple masters can be simultaneously involved in a transfer to the same address, or device, then the design of the system must be done in such a way that arbitration between:

   - A repeated Start condition and a data bit does not occur.
   - A Stop condition and a data bit does not occur.
   - A repeated Start condition and a Stop condition does not occur.

   An example of this situation occurs if one master writes 1 byte while another master writes 2 bytes to the same device.
9. IIC0SDA is shared with GPIO bit 26.

   Sharing is controlled by the SDR Register SDR0_PFC4 bit 0.

31. GPIO Operations

This chapter describes the General Purpose I/O (GPIO) controllers attached to the on-chip peripheral bus (OPB) of the PPC440EPx/GRx. The GPIO controllers provide flexible control of multiplexed I/Os selectable under program control. Each of the I/Os is multiplexed with other signals to reduce the quantity of I/O pins needed on the PPC440EPx/GRx package.

31.1 Overview

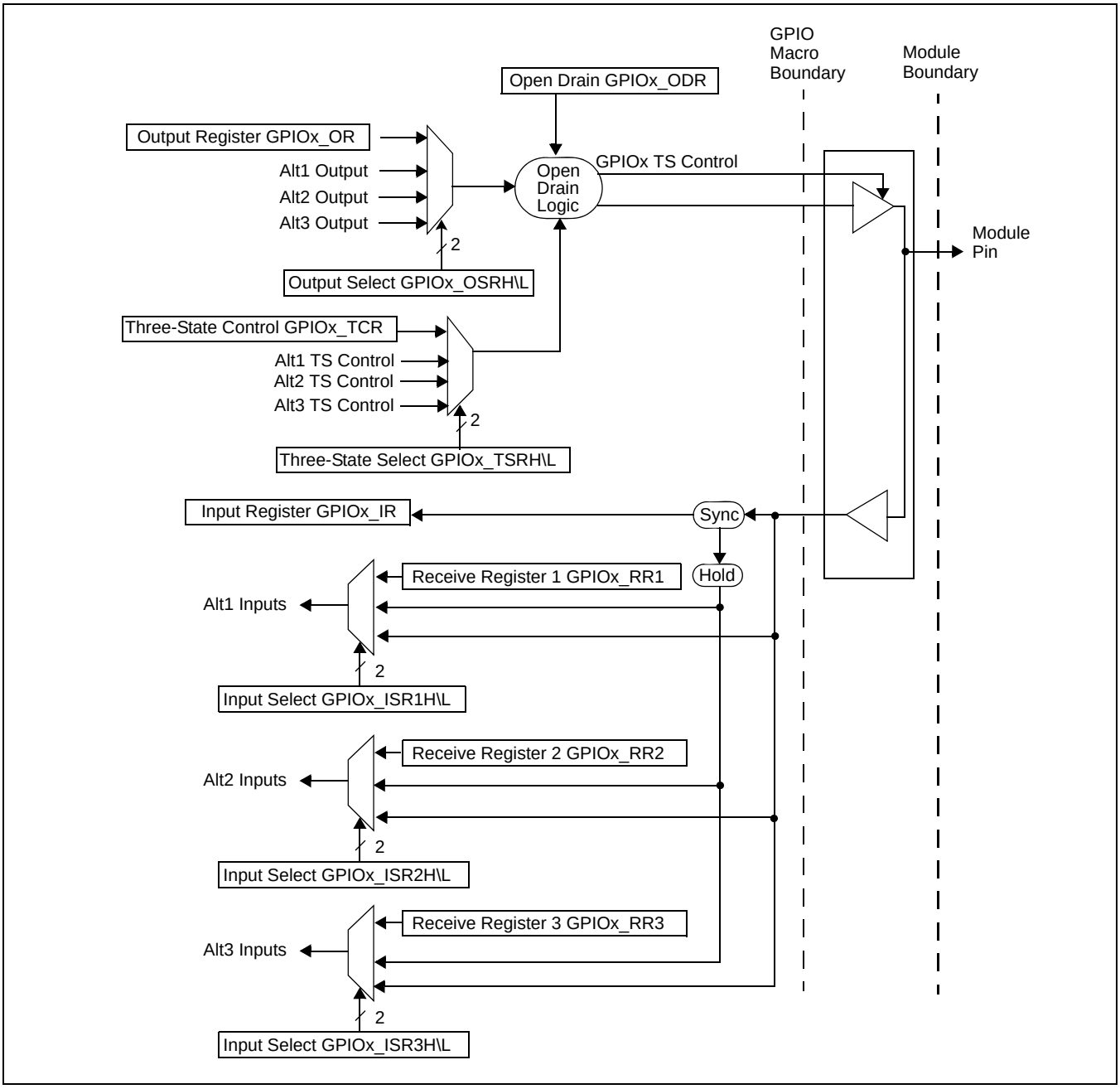
The PPC440EPx/GRx has two 32-bit GPIO controllers: GPIO0 and GPIO1. GPIO0 provides 32 user-programmable external signals, multiplexed with system-related signal groups including trace outputs, external interrupt inputs, chip selects, DMA, EBC, ZMII, RGMII, and USB2D interface signals. GPIO1 provides another 32 programmable signals, multiplexed with USB2D, UART0, UART1, UART2, DMA and UIC interfaces. When information applies to both GPIO controllers in the PPC440EPx/GRx, references are made generic to avoid repetition: GPIOx, not GPIO0 and GPIO1.

I/O signals maintain a bit-for-bit correspondence with both Input and Output Register bits. For example, a signal output to pin GPIO1_Out[6] can be sourced by GPIO1 Output Register bit 6 or by either of two alternative output signals, UART0_DTR and UART2_Tx.

Signals sourced to all GPIO outputs are selected by the Output Select Register (GPIOx_OSR), while output buffer operation (three-state output, open-drain output) is controlled by the Open Drain Register (GPIOx_ODR) and the Three-State Control Register (GPIOx_TCR).

Figure 31-1 shows data flow between one module pin and the alternative sources and destinations to which the module pin is connected. For example, a signal input on a GPIO pin is captured in the corresponding bit of the GPIO Input Register, as well as multiplexed to one or two other internal functional connections. The multiplexed signals are generically referred to as Alt1, Alt2 or Alt3 inputs and outputs.

Figure 31-1. GPIO Dataflow and Configuration Registers



31.2 Features

Each of the GPIO Controllers has the following features:

- Direct control of all GPIO controller functions from registers programmed via memory-mapped addresses
  - Each output can be selected from one of four sources.
  - Each output three-state control can be selected from one of four sources.
  - Selection of the input source for the Alternate 1 input
  - Selection of the input source for the Alternate 2 input
  - Selection of the input source for the Alternate 3 input
  - Settable register as one of the possible inputs for the Alternate 1 input
  - Settable register as one of the possible inputs for the Alternate 2 input
  - Settable register as one of the possible inputs for the Alternate 3 input
- Control of 32 bidirectional GPIO module pins
  - Each GPIO output can be set from the corresponding Output Register bit.
  - Each GPIO output has programmable three-state control.
  - Each GPIO output can also be programmed to emulate an open drain output.
  - Each GPIO input is observable from the corresponding Input Register bit.

31.3 Clock and Power Management

The GPIO controllers support clock and power management. Unconditional Sleep (Class 1) power management is implemented for each controller, and is enabled by setting the corresponding CPM0_ER[GPIOx] bit, as shown in *CPM Enable Registers (CPM0_ER and CPM1_ER)* on page 291.

31.4 GPIO Registers

When an I/O is used as GPIO it is controlled by the corresponding bit in the Output Register GPIOx_OR, the Three-State Control Register GPIOx_TCR, the Open Drain Register GPIOx_ODR, and the Input Register GPIOx_IR. Whether an I/O is used as a GPIO or a functional output is selected using the Output Select Register pair GPIOx_OSRH and GPIOx_OSRL. The source of the three-state control is selected using the Three-State Select Register pair GPIOx_TSRH and GPIOx_TSRL. When an alternate input is used, the source of the alternate input is selected using the Input Select Register GPIOx_ISRnL and GPIOx_ISRnH. Note that there are two input select register pairs; one pair for Alternate 1 functions and one pair for Alternate 2 functions. The controller also contains Receive Registers GPIOx_RRx. A detailed description of the function of each of the registers is contained in *Detailed Register Descriptions* on page 799.

Table 31-1 contains a summary of the GPIO registers.

Table 31-1. GPIO Register Summary

Mnemonic	Address	Access	Description	Page
GPIO0_OR	0x1 EF60 0B00	R/W	GPIO0 Output	799
GPIO0_TCR	0x1 EF60 0B04	R/W	GPIO0 Three-State Control Register	799
GPIO0_OSRL	0x1 EF60 0B08	R/W	GPIO0 Output Select Register Low	799

Table 31-1. GPIO Register Summary (continued)

Mnemonic	Address	Access	Description	Page
GPIO0_OSRH	0x1 EF60 0B0C	R/W	GPIO0 Output Select Register High	799
GPIO0_TSRL	0x1 EF60 0B10	R/W	GPIO0 Three-State Select Register Low	800
GPIO0_TSRH	0x1 EF60 0B14	R/W	GPIO0 Three-State Select Register High	800
GPIO0_ODR	0x1 EF60 0B18	R/W	GPIO0 Open Drain Register	800
GPIO0_IR	0x1 EF60 0B1C	R	GPIO0 Input Register	801
GPIO0_RR1	0x1 EF60 0B20	R/W	GPIO0 Receive Register 1	802
GPIO0_RR2	0x1 EF60 0B24	R/W	GPIO0 Receive Register 2	
GPIO0_RR3	0x1 EF60 0B28	R/W	GPIO0 Receive Register 3	802
GPIO0_ISR1L	0x1 EF60 0B30	R/W	GPIO0 Input Select Register 1 Low	801
GPIO0_ISR1H	0x1 EF60 0B34	R/W	GPIO0 Input Select Register 1 High	801
GPIO0_ISR2L	0x1 EF60 0B38	R/W	GPIO0 Input Select Register 2 Low	801
GPIO0_ISR2H	0x1 EF60 0B3C	R/W	GPIO0 Input Select Register 2 High	801
GPIO0_ISR3L	0x1 EF60 0B40	R/W	GPIO0 Input Select Register 3 Low	801
GPIO0_ISR3H	0x1 EF60 0B44	R/W	GPIO0 Input Select Register 3 High	801
GPIO1_OR	0x1 EF60 0C00	R/W	GPIO1 Output Register	799
GPIO1_TCR	0x1 EF60 0C04	R/W	GPIO1 Three-State Control Register	799
GPIO1_OSRL	0x1 EF60 0C08	R/W	GPIO1 Output Select Register Low	799
GPIO1_OSRH	0x1 EF60 0C0C	R/W	GPIO1 Output Select Register High	799
GPIO1_TSRL	0x1 EF60 0C10	R/W	GPIO1 Three-State Select Register Low	800
GPIO1_TSRH	0x1 EF60 0C14	R/W	GPIO1 Three-State Select Register High	800
GPIO1_ODR	0x1 EF60 0C18	R/W	GPIO1 Open Drain Register	800
GPIO1_IR	0x1 EF60 0C1C	R	GPIO1 Input Register	801
GPIO1_RR1	0x1 EF60 0C20	R/W	GPIO1 Receive Register 1	802
GPIO1_RR2	0x1 EF60 0C24	R/W	GPIO1 Receive Register 2	
GPIO1_RR3	0x1 EF60 0C28	R/W	GPIO1 Receive Register 3	802
GPIO1_ISR1L	0x1 EF60 0C30	R/W	GPIO1 Input Select Register 1 Low	801
GPIO1_ISR1H	0x1 EF60 0C34	R/W	GPIO1 Input Select Register 1 High	801
GPIO1_ISR2L	0x1 EF60 0C38	R/W	GPIO1 Input Select Register 2 Low	801
GPIO1_ISR2H	0x1 EF60 0C3C	R/W	GPIO1 Input Select Register 2 High	801
GPIO1_ISR3L	0x1 EF60 0C40	R/W	GPIO1 Input Select Register 3 Low	801
GPIO1_ISR3H	0x1 EF60 0C44	R/W	GPIO1 Input Select Register 3 High	801

Notes: 1. All GPIO registers are memory-mapped and accessed using load/store instructions at the register address. All GPIO registers are aligned on word boundaries. The input and output select registers are also aligned on double word boundaries.

2. Low register set controls GPIOx I/O signals 0 to 15. High register set controls GPIOx I/O signals 16 to 31.

31.5 GPIO Register Reset Values

When a system reset occurs, all register bits in the GPIO controllers, except GPIOx_IR, are reset to 0. All outputs are placed in high impedance. GPIOx_IR is not reset because it is synchronized to the OPB clock and always tracks the state of the I/O pins.

31.6 Detailed Register Descriptions

The following sections provide detailed descriptions of the GPIO controller registers. The two GPIO controllers are attached to the OPB bus. The GPIOx_IR register is read-only; all other registers are both read- and write-accessible.

31.6.1 GPIO Output Register (GPIOx_OR)

When a bit in one of the two GPIO controllers is used as a GPIO and specifically as an output, the state of the output is controlled by the value in the GPIOx Output Register, GPIOx_OR. Whether the setting of the bit in the GPIOx_OR is visible on the I/O pin is a function of the settings in the GPIOx_TCR, GPIOx_ODR, GPIOx_OSRH, GPIOx_OSRL, GPIOx_TSRH, and GPIOx_TSRL registers.

Figure 31-2. GPIO Output Register (GPIOx_OR)

0:31		GPIO0_OR register bits	

31.6.2 GPIO Three-State Control Register (GPIOx_TCR)

The GPIOx_TCR register is one source for three-stating a corresponding output. For each bit in the GPIOx_TCR Register to control the corresponding output, the appropriate two bits in the GPIO Three-State Select Register High and Low (GPIOx_TSRH, GPIOx_TSRL) must be programmed to 0b00. If the GPIOx_TCR register is selected; setting a bit to 1 in GPIOx_TCR enables the associated output driver. Clearing the bit to 0 forces the corresponding output into high impedance state. When the same bit is also set in GPIOx_ODR, the output emulates an open-drain output, regardless of the bit setting in GPIOx_TCR.

Figure 31-3. GPIO Three-State Register (GPIOx_TCR)

0:31		GPIO0_TCR register bits	

31.6.3 GPIO Output Select Registers (GPIOx_OSRH, GPIOx_OSRL)

The GPIOx_OSR register pair (GPIOx_OSRH, GPIOx_OSRL) determines what signal source is sent to the output pin. For each output there can be up to four sources of output value. Two bits in the GPIOx_OSRH/L register pair are needed to select each output. This requires a total of 64 bits to control the 32 output bits in one GPIO controller. The 64 bits are made up of two 32 bit registers. *Table 31-2* shows how these two bits control the selection of the signal connected to that GPIO Out pin.

Note: The correspondence between GPIOx_OSRH/L and pins 0:63 is summarized in *Section 31.7* on page 803.

Table 31-2. GPIO Output Signal Selection

GPIOx_OSRH/L Bits	GPIO_Out Signal Source
00	GPIOx_OR
01	Alt1 output source
10	Alt2 output source
11	Alt3 output source

31.6.4 GPIO Three-State Select Registers (GPIOx_TSRH, GPIOx_TSRL)

The GPIOx_TSR register pair (GPIOx_TSRH, GPIOx_TSRL) determines what signal source is used for the three-state control input for the GPIO output signals.

For each output there can be two sources of three-state control. Two bits in the GPIOx_TSRH/L register pair are needed for each output. This requires a total of 64 bits to control the 32 output bits in one GPIO controller. The 64 bits are made up of two 32 bit registers.

The three-state selection for a pin requires the same combination as the output select register selection. That is, if the Alt2 is selected then the Alt2 three-state control will be selected. *Table 31-3* shows how these two bits control the selection of the signal connected to that GPIO_Out pin.

Note: The correspondence between GPIOx_TSRH/L and pins 0:63 is summarized in *Section 31.7* on page 803.

Table 31-3. GPIO Three-State Selection

GPIOx_TSRH/L Bits	GPIO Three-State Control Source
00	GPIOx_TCR
01	Alt1 three-state source
10	Alt2 three-state source
11	Alt3 three-state source

31.6.5 GPIO Open Drain Register (GPIOx_ODR)

The GPIOx_ODR configures the module I/O three-state driver to emulate open drain drivers on a bit-by-bit basis. *Table 31-4* shows the function of the GPIOx_ODR register and its interaction with the three-state control signal and output signal. While the GPIOx_ODR register can control Alternate 1, 2 and 3 outputs, as

well as alternate three-state control signals, this feature is not used in the PPC440EPx/GRx. For this reason the registers listed in the table are used when a bit is used as a GPIO.

Table 31-4. GPIO0_ODR Control Settings

GPIOx_ODR bit	GPIOx_TCR bit	GPIOx_OR bit	State of Module pin
0	0	x	Forced to high impedance
0	1	0	Driving 0
0	1	1	Driving 1
1	x	0	Driving 0
1	x	1	Forced to high impedance

Figure 31-4. GPIO Open Drain Register (GPIOx_ODR)

0:31		GPIO0_ODR register bits	

31.6.6 GPIO Input Register (GPIOx_IR)

The state of each bit in the GPIOx_IR Register reflects the corresponding GPIO controller input signal. All input signals are synchronized to OPBClk before being stored in the GPIOx_IR Register. The GPIOx_IR register is read-only and does not change during a read access. To receive valid data as input from a module pin, the three-state output attached to that module pin is first placed in high impedance by setting the corresponding three-state control bit in the GPIOx_TCR register.

Figure 31-5. GPIO Input Register (GPIOx_IR)

0:31		GPIO register bits	

31.6.7 GPIO Input Select Registers (GPIOx_ISRnH, GPIOx_ISRnL)

The GPIOx_ISR1H/L, GPIOx_ISR2H/L and GPIOx_ISR3H/L registers determine what signal source is used as the input signals. For each Altn input there can be up to three sources of input value. Two bits in the GPIOx_ISR1H/L register pair are needed for each Alt1 input, two bits in the GPIOx_ISR2H/L register pair are needed for each Alt2 input, and two bits in the GPIOx_ISR3H/L register pair are needed for each Alt3 input. This requires a total of 64 bits to control the 32 Alt1 input bits, 64 bits to control the 32 Alt2 input bits in one GPIO controller, and 64 bits to control the 32 Alt3 input bits. The 64 bits are made up of two 32 bit registers. *Table 31-5* shows how these two bits control the selection of the input. For normal operation the only setting used is 01 which selects the pin input.

Note: The correspondence between GPIOx_ISRnH/L and pins 0:63 is summarized in *Section 31.7* on page 803.

Table 31-5. GPIO Alternate Input Signal Selection

GPIOx_ISRnH/L Bits	Altn Input Signal
00	GPIO Receive Register (GPIOx_RRn)
01	Input from GPIO pin (not synchronized) - configuration used by Altn signals
10	Input from GPIO pin (synchronized and latched)
11	Reserved

31.6.8 GPIO Receive Registers (GPIOx_RR1, GPIOx_RR3)

For normal operation the receive registers are not used, as noted in later sections on selecting the Alt1, Alt2 and Alt3 signals. They may however be useful during software debug. There are times when a programmer would like to control input values without having to determine how to cause the system to produce the desired value at an input. By programming the desired input bit in GPIOx_RR1 for Alt1 inputs, GPIOx_RR2 for Alt2 inputs, GPIOx_RR3 for Alt3 inputs and then setting the appropriate two bits, in the associated GPIOx_ISRnH/L register pair to 00, a desired input value can be set. Note that these registers only affect the alternate inputs, not the GPIOx_IR register.

31.6.9 Control of GPIO Signals 49–63

GPIO49:63 signals are multiplexed with CPU trace signals. In order to use these signals in GPIO mode, the SDR0_PFC0 register must be properly initialized (*Section 7.4.2.5* on page 177). There are no alternates inputs or outputs for these pins. They are controlled by the GPIO1_xxxH registers.

31.7 GPIO Signal Multiplexing

The following tables describe the multiplexed signal assignments for the GPIO0 and GPIO1controllers. All of the GPIO signals are multiplexed with one or more additional signals. Each of the package pins on which these signals appear has one of the indicated signal (the default or primary signal) connected following a reset of the chip. The GPIO signals are not the primary signal for any pin. The primary signal is the signal in the Alternate 1 column.

Table 31-6. Alternate 1 Configuration for GPIO00–GPIO15

GPIO Signal	Alternate 1 Signal	I/O	GPIO0_TCR		GPIO0_OSRL		GPIO0_TSRL		GPIO0_ISR1L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO00	PerAddr07[1]	I/O	0	x	0:1	01	0:1	01	0:1	01
GPIO01	PerAddr06[1]	I/O	1	x	2:3	01	2:3	01	2:3	01
GPIO02	PerAddr05[1]	I/O	2	x	4:5	01	4:5	01	4:5	01
GPIO03	PerAddr04[1]	I/O	3	x	6:7	01	6:7	01	6:7	01
GPIO04	PerAddr03[1]	I/O	4	x	8:9	01	8:9	01	8:9	01
GPIO05	PerAddr02[1]	I/O	5	x	10:11	01	10:11	01	10:11	01
GPIO06	PerCS1, $\overline{\text{NFCE1}}$[2]	O	6	x	12:13	01	12:13	01	12:13	xx
GPIO07	PerCS2, $\overline{\text{NFCE2}}$[2]	O	7	x	14:15	01	14:15	01	14:15	xx
GPIO08	PerCS3, $\overline{\text{NFCE3}}$[2]	O	8	x	16:17	01	16:17	01	16:17	xx
GPIO09	PerCS4	O	9	x	18:19	01	18:19	01	18:19	xx
GPIO10	PerCS5	O	10	x	20:21	01	20:21	01	20:21	xx
GPIO11	PerErr	I	11	x	22:23	xx	22:23	01	22:23	01

Notes: 1. PerAddr02:07 can be configured as an output when not using the EBM.
2. SDR0_CUST0[NCG] controls the Alternate 1 signal multiplexing of the PerCS1:3/NFCE1:3 signals.

Table 31-7. Alternate 1 Configuration for GPIO16–GPIO31

GPIO Signal	Alternate 1 Signal	I/O	GPIO0_TCR		GPIO0_OSRH		GPIO0_TSRH		GPIO0_ISR1H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO16	GMCTxD4, GMC1TxD0	O	16	1	0:1	01	0:1	00	0:1	xx
GPIO17	GMCTxD5, GMC1TxD1	O	17	1	2:3	01	2:3	00	2:3	xx
GPIO18	GMCTxD6, GMC1TxD2	O	18	1	4:5	01	4:5	00	4:5	xx
GPIO19	GMCTxD7, GMC1TxD3	O	19	1	6:7	01	6:7	00	6:7	xx
GPIO20	RejectPkt0	I	20	0	8:9	xx	8:9	00	8:9	01
GPIO21	RejectPkt1	I	21	0	10:11	XX	10:11	00	10:11	01
GPIO23	SCPDO	O	23	x	14:15	01	14:15	01	14:15	xx
GPIO24	GMCTxD2, GMC0TxD2	O	24	1	16:17	01	16:17	00	16:17	xx
GPIO25	GMCTxD3, GMC0TxD3	O	25	1	18:19	01	18:19	00	18:19	xx
GPIO27	ExtReq	I	27	0	22:23	xx	22:23	00	22:23	01
GPIO28	See *Table 31-11*.									
GPIO29	HoldAck	O	29	x	26:27	01	26:27	01	26:27	xx
GPIO30	ExtAck	O	30	x	28:29	01	28:29	01	28:29	xx
GPIO31	BusReq	O	31	x	30:31	01	30:31	01	30:31	xx

Table 31-8. Alternate 1 Configuration for GPIO32–GPIO47

GPIO Signal	Alternate 1 Signal	I/O	GPIO1_TCR		GPIO1_OSRL		GPIO1_TSRL		GPIO1_ISR1L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO32	USB2OM0 *(PPC440EPx only)*	O	0	1	0:1	01	0:1	00	0:1	xx
GPIO33	USB2OM1 *(PPC440EPx only)*	O	1	1	2:3	01	2:3	00	2:3	xx
GPIO34	$\overline{\text{UART0\_DCD}}$	I	2	0	4:5	xx	4:5	00	4:5	01
GPIO35	$\overline{\text{UART0\_DSR}}$	I	3	0	6:7	xx	6:7	00	6:7	01
GPIO36	$\overline{\text{UART0\_CTS}}$, $\overline{\text{UART0\_DSR}}$[1]	I	4	0	8:9	xx	8:9	00	8:9	01
GPIO37	$\overline{\text{UART0\_RTS}}$, $\overline{\text{UART0\_DTR}}$[1]	O	5	1	10:11	01	10:11	00	10:11	xx
GPIO38	$\overline{\text{UART0\_DTR}}$	O	6	1	12:13	01	12:13	00	12:13	xx
GPIO39	$\overline{\text{UART0\_RI}}$	I	7	0	14:15	xx	14:15	00	14:15	01
GPIO40	IRQ0	I	8	0	16:17	xx	16:17	00	16:17	01
GPIO41	IRQ1	I	9	0	18:19	xx	18:19	00	18:19	01
GPIO42	IRQ2	I	10	0	20:21	xx	20:21	00	20:21	01
GPIO43	IRQ3	I	11	0	22:23	xx	22:23	00	22:23	01
GPIO44	IRQ4	I	12	0	24:25	xx	24:25	00	24:25	01
GPIO45	IRQ6	I	13	0	26:27	xx	26:27	00	26:27	01
GPIO46	IRQ7	I	14	0	28:29	xx	28:29	00	28:29	01
GPIO47	IRQ8	I	15	0	30:31	xx	30:31	00	30:31	01

Notes: 1. SDR0_PFC1[U0ME] specifies for the UART0 in 4 wire mode (SDR0_PFC1[U0IM] = 1) which control signals are used: DSR/DTR or CTS/RTS.

Table 31-9. Alternate 1 Configuration for GPIO48

GPIO Signal	Alternate 1 Signal	I/O	GPIO1_TCR		GPIO1_OSRH		GPIO1_TSRH		GPIO1_ISR1H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO48	IRQ9	I	16	0	0:1	xx	0:1	00	0:1	01

Table 31-10. Alternate 2 Configuration for GPIO00–GPIO05

GPIO Signal	Alternate 2 Signal	I/O	GPIO0_TCR		GPIO0_OSRL		GPIO0_TSRL		GPIO0_ISR2L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO00	DMAReq2	I	0	0	0:1	xx	0:1	00	0:1	01
GPIO01	DMAAck2	O	1	1	2:3	10	2:3	00	2:3	xx
GPIO02	EOT2/TC2	I/O	2	x	4:5	10	4:5	10	4:5	01
GPIO03	DMAReq3	I	3	0	6:7	xx	6:7	00	6:7	01
GPIO04	DMAAck3	O	4	1	8:9	10	8:9	00	8:9	xx
GPIO05	EOT3/TC3	I/O	5	x	10:11	10	10:11	10	10:11	01

Table 31-11. Alternate 2 Configuration for GPIO27–GPIO31

GPIO Signal	Alternate 2 Signal	I/O	GPIO0_TCR		GPIO0_OSRH		GPIO0_TSRH		GPIO0_ISR2H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO27	USB2RxErr *(PPC440EPx only)*	I	27	0	22:23	xx	22:23	00	22:23	01
GPIO28	USB2TxVal *(PPC440EPx only)*	O	28	1	24:25	10	24:25	00	24:25	xx
GPIO29	USB2Susp *(PPC440EPx only)*	O	29	1	26:27	10	26:27	00	26:27	xx
GPIO30	USB2XcvrSel *(PPC440EPx only)*	O	30	1	28:29	10	28:29	00	28:29	xx
GPIO31	USB2TermSel *(PPC440EPx only)*	O	31	1	30:31	10	30:31	00	30:31	xx

Table 31-12. Alternate 2 Configuration for GPIO32–GPIO47

GPIO Signal	Alternate 2 Signal	I/O	GPIO1_TCR		GPIO1_OSRL		GPIO1_TSRL		GPIO1_ISR2L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO32	PerDataPar2	I/O	0	x	0:1	10	0:1	10	0:1	01
GPIO33	PerDataPar3	I/O	1	x	2:3	10	2:3	10	2:3	01
GPIO34	$\overline{\text{UART2\_CTS}}$, $\overline{\text{UART2\_DSR}}$	I	2	0	4:5	xx	4:5	00	4:5	01
GPIO35	$\overline{\text{UART2\_RTS}}$, $\overline{\text{UART2\_DTR}}$	O	3	1	6:7	10	6:7	00	6:7	xx
GPIO36	PerDataPar0	I/O	4	x	8:9	10	8:9	10	8:9	01
GPIO37	PerDataPar1	I/O	5	x	10:11	10	10:11	10	10:11	01
GPIO38	UART1_Tx	O	6	1	12:13	10	12:13	00	12:13	xx
GPIO39	UART1_Rx	I	7	0	14:15	xx	14:15	00	14:15	01
GPIO40-GPIO43	Alternate 1 signal only. See *Table 31-9*.									
GPIO44	DMAAck1	O	12	1	24:25	10	24:25	00	24:25	xx
GPIO45	EOT1/TC1	I/O	13	x	26:27	10	26:27	10	26:27	01
GPIO46	DMAReq0	I	14	0	28:29	xx	28:29	00	28:29	01
GPIO47	DMAAck0	O	15	1	30:31	10	30:31	00	30:31	xx

Table 31-13. Alternate 2 Configuration for GPIO48

GPIO Signal	Alternate 2 Signal	I/O	GPIO1_TCR		GPIO1_OSRH		GPIO1_TSRH		GPIO1_ISR2H	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO48	EOT0/TC0	I/O	16	x	0:1	10	0:1	10	0:1	01

Table 31-14. Alternate 3 Configuration for GPIO34–GPIO37

GPIO Signal	Alternate 3 Signal	I/O	GPIO1_TCR		GPIO1_OSRL		GPIO1_TSRL		GPIO1_ISR3L	
			Bit	Value	Bit	Value	Bit	Value	Bit	Value
GPIO34	UART2_Tx	O	2	1	4:5	11	4:5	00	4:5	xx
GPIO35	UART2_Rx	I	3	0	6:7	xx	6:7	00	6:7	01
GPIO36	UART3_Rx	I	4	0	8:9	xx	8:9	00	8:9	01
GPIO37	UART3_Tx	O	5	1	10:11	11	10:11	00	10:11	xx

32. Universal Serial Bus Interfaces

The following sections provide detailed information for the Universal Serial Bus (USB) interfaces implemented in the PPC440EPX chip.

Note: USB Host and Device interfaces are supported by the PPC440EPx only.

32.1 USB Overview

The USB is an open serial data bus standard that allows devices to be attached to a computer through a single, processor-attached peripheral to which the devices are chained or attached in a star topology. The PPC440EPx USB2.0 interface has the following features:

- USB 2.0 Host with internal PHY
  - Fully compliant to the following specifications:
    - Universal Serial Bus Specification, Revision 2.0.
    - Enhanced Host Controller Interface (EHCI) Specification for USB, Revision 1.0.
    - Open Host Controller Interface (OHCI) Specification for USB, Revision 1.0a.
  - One USB port provided through a UTMI transceiver interface connected to the internal USB 2.0 Physical Layer (Single port USB 2.0 Physical Layer UTMI+ version 1.05).
  - One USB EHCI high speed (480Mbps) host controller with 1024-byte packet buffer.
  - One USB OHCI full/low speed (12Mbps/1.5Mbps) host controller.
  - Host controller does not support high-bandwidth isochronous transfers
  - Independent OPB master and slave ports which run asynchronously to the core USB clocks.
  - USB OHCI/EHCI registers and data structures implemented in Big Endian format.
- USB 2.0 Device UTMI or USB 2.0 Device with internal PHY (excluding USB 2.0 Host function)
  - Device support provides six end points (three IN, three OUT)
  - 8192B FIFO by endpoint (supports high-bandwidth isochronous transfers, double buffering of 1024B packets)
  - FIFOs are not shared between IN and OUT endpoints
  - Two USB 2.0 Device end points have DMA dedicated channels (DMA-to-PLB 128)

32.1.1 References

Users of this interface should refer to the following documents and standards:

- USB Version 2.0 Specification
- USB Enhanced Host Controller Interface Specification
- USB Open Host Controller Interface Specification
- *Inventra MUSBHSFC-OPB Programmer's Guide*, PG-40131.005-FC, 07/04 or later
- OPB Architecture Version 1.4 Specification
- *Intel High Speed USB Platform Design , Revision 1.0*

32.2 USB 2.0 Configuration

The following registers control the configuration of the USB interface. All of these registers are 32 bits.

.

Table 32-1. USB Configuration Registers

Register Name	Description	Offset	Access	Page
SDR0_USB2PHY0CR	USB2PHY0 Control Register	0x4103	R/W	810
SDR0_USB2H0ST	USB2 Host Status Register	0x4600	R	811
SDR0_USB2H0CR	USB2 Host Control Register	0x0340	R/W	811
SDR0_USB2D0CR	USB2 Device Control Register	0x0320	R/W	812

32.2.1 USB2PHY0 Control Register (SDR0_USB2PHY0CR)

This register is used to configure the internal USB2.0 PHY.

Offset: 0x4103

Access: R/W

Figure 32-1. USB2PHY0 Control Register (SDR0_USB2PHY0CR)

Bit	Mnemonic	Description	Comments
0:1	CLKSEL	Select reference clock frequency 00 Reserved 01 48MHz 10 Reserved 11 Reserved	Reset value: 01
2	XOCLK	Select external input clock source 0 Oscillator 1 Crystal	Reset value: 1
3	XOREF	Select reference clock source 0 Reserved 1 USB2PHY0	Reset value: 1
4	PWRSAV	Select PHY power-save mode 0 Non-power-save mode. Valid in all modes and speeds of operation. 1 Power-save mode. Valid only for full-speed operation.	CLKSEL must be 01 (48MHz) Reset value: 0
5		Reserved	
6		Reserved	
7	WDINT	PHY UTMI data width and clock select 0 8-bit data/60MHz 1 16-bit data/30MHz	When connected to USB2D0, only the 8-bit/ 60MHz connection is supported. Reset value: 1
8	DVBUS	VBus detect (Device mode only) 0 Pull-up resistance on D+ is disabled 1 Pull-up resistance on D+ is enabled based on the speed of operation	Controls the pull-up resistance on the D+ line in Device mode only. Reset value: 0

Figure 32-1. USB2PHY0 Control Register (SDR0_USB2PHY0CR) (continued)			
Bit	**Mnemonic**	**Description**	**Comments**
9	DWNSTR	Select downstream port mode 0 Device 1 Host	Sets the PHY to Host or Device mode. Reset value: 1
10		Reserved	
11	UTMICN	UTMI interface connection 0 Connect USB2PHY0 to USB2D0. 1 Connect USB2PHY0 to USB2H0.	Sets the PHY UTMI interface connection. Reset value: 1
12:31		Reserved	

32.2.2 USB2 Host Status Register (SDR0_USB2H0ST)

This register contains USB2.0 Host status information from EHCI and OHCI. These status bits are also available in EHCI and OHCI internal registers.

Offset: 0x4600

Access: Read only

Figure 32-2. USB2 Host Status Register (SDR0_USB2H0ST)			
Bit	**Mnemonic**	**Description**	**Comments**
0:22		Reserved	
23	ORWE	OHCI Remote Wakeup Enabled	Indicates the state of the OHCI HcControl RWE bit.
24	ODRWE	OHCI Device Remote Wakeup Enabled	Indicates the state of the OHCI HCRhStatus DRWE bit.
25	OSUS	OHCI port suspend status 0 Active 1 Suspended	Indicates the state of the OHCI port.
26	IAA	Interrupt on Async Advance	Indicates the state of the EHCI IAA status bit.
27	HSE	Host System Error	Indicates the state of the EHCI HSE status bit.
28	FLR	Frame List Rollover	Indicates the state of the EHCI FLR status bit.
29	PCD	Port Change Detect	Indicates the state of the EHCI PCD status bit.
30	ERRINT	USB Error Interrupt	Indicates the state of the EHCI ERRINT status bit.
31	USBINT	USB Interrupt	Indicates the state of the EHCI USBINT status bit.

32.2.3 USB2 Host Control Register (SDR0_USB2H0CR)

This register is used to configure the USB2.0 Host Controller.

Offset: 0x0340

Access: R/W

Figure 32-3. USB2 Host Control Register (SDR0_USB2H0CR)

Bit	Mnemonic	Description	Comments
0:24		Reserved	
25:30	EFLADJ	EHCI Frame Length Adjustment	This feature adjusts any offset from the clock source that drives the EHCI host controller's Start of Frame (SOF) counter. The SOF cycle time (number of SOF counter clock periods to generate an SOF microframe length) is equal to 59,488 plus the value represented by this input signal. The following shows the actual microframe bit times resulting from each possible value tied to the SS_FLADJ_VAL_HOST[5:0] input bus: Frame Length in decimal bit times/ SS_FLADJ_VAL_HOST[5:0] 59488 / 0x00 59504 / 0x01 59520 / 0x02 . . . 59984 / 0x1F 60000 / 0x20 . . . 60496 / 0x3F **Note:** This input should only be modified when EHCI_USBSTS[HCHalted] = 1; otherwise, the host controller will yield undefined results. It should not be reprogrammed by the USB system software unless the default or BIOS programmed values are incorrect or the system is restoring the register while returning from a suspended state. Reset value: 0x20
31	WDINT	Host UTMI Word Interface 0 8-bit/60MHz 1 16-bit/30MHz	For a PHY to Host connection, this bit must be coherent with SDR0_USB2PHY0CR[WDINT] . Reset value: 1

32.2.4 USB2 Device Control Register (SDR0_USB2D0CR)

This register is used to configure the USB2.0 Device Controller (I/O selection between UTMI interface and EBC 32-bit).

Offset: 0x0320

Access: R/W

Figure 32-4. USB2 Device Control Register (SDR0_USB2D0CR)

Bit	Mnemonic	Description	Comments
0:28		Reserved	
29	USBEBC	Select USB 2.0 Device or EBC. 0 EBC 32-bit selected 1 USB2 Device selected	USB2EBC controls the function of signals: PerData16:23[USB2DO0:7} PerData24:31[USB2DI0:7]
30:31		Reserved	

32.3 USB 2.0 PHY Sharing

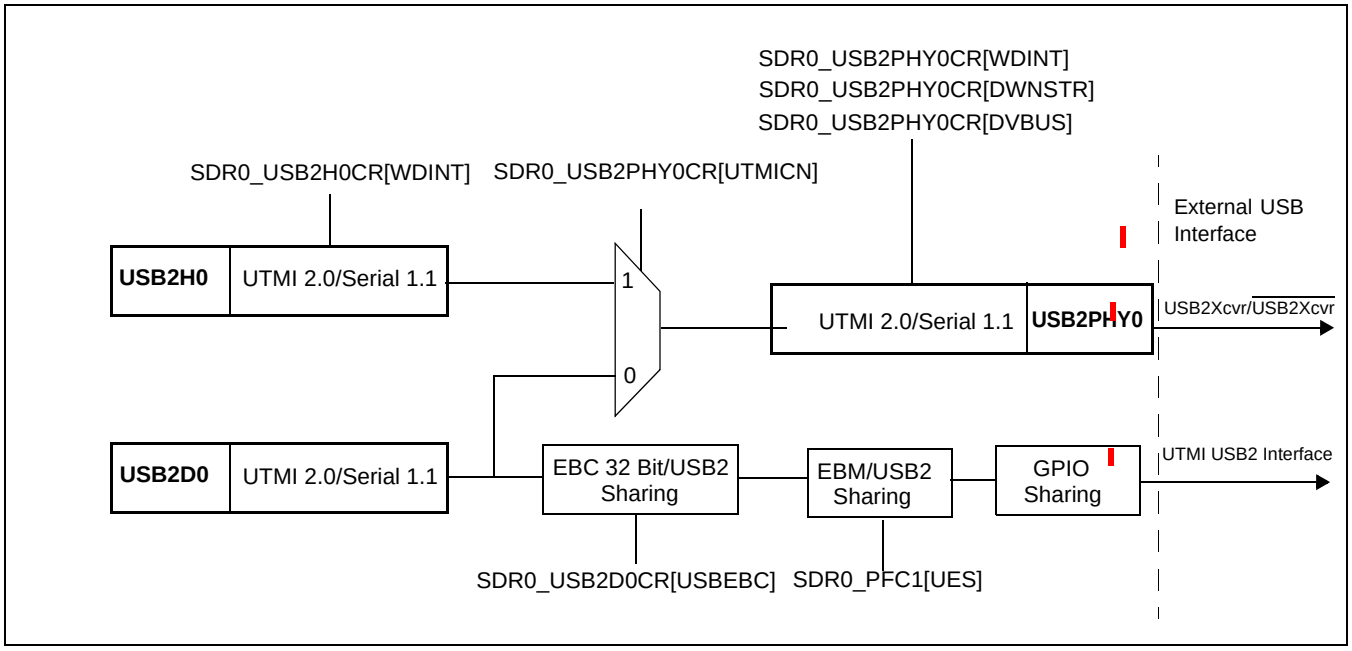
The PPC440EPx contains an internal USB2.0 PHY, USB2PHY0, that provides an interface between the USB cable and a USB 2.0 Host or Device controller.

The Host controller, USB2H0, can only be interfaced through the USB2.0 PHY.

The Device controller, USB2D0, can be connected to either the USB2PHY0 or to the external UTMI interface. When connected to the USB2PHY0, the Host controller, USBH0, cannot be used. When connected to the external UTMI interface, the USB2D0 and USB2H0 controllers can both be used.

The following figure illustrates the sharing of USB2PHY0 between USB2H0 and USBD0:

Figure 32-5. USB 2.0 PHY Configuration



32.3.1 USB2PHY0 Connected to USB2H0

To connect USB2PHY0 to USB2H0 configure as follows:

- SDR0_USB2PHY0CR[UTMICN] = 1
- SDR0_USB2H0CR[WDINT] = 1 and SDR0_USB2PHY0CR[WDINT] = 1
- SDR0_USB2PHY0CR[DWNSTR] = 1
- SDR0_USB2PHY0CR[DVBUS] = 0

To connect the external UTMI interface to USB2D0 configure as follows:

- SDR0_USB2D0CR[USBEBC] = 1
- SDR0_PFC1[UES] = 0

  To configure signals shared with GPIO, see *GPIO Operations* on page 795.
  The external UTMI interface is shared with the EBC interface.

32.3.2 USB2PHY0 Connected to USB2D0

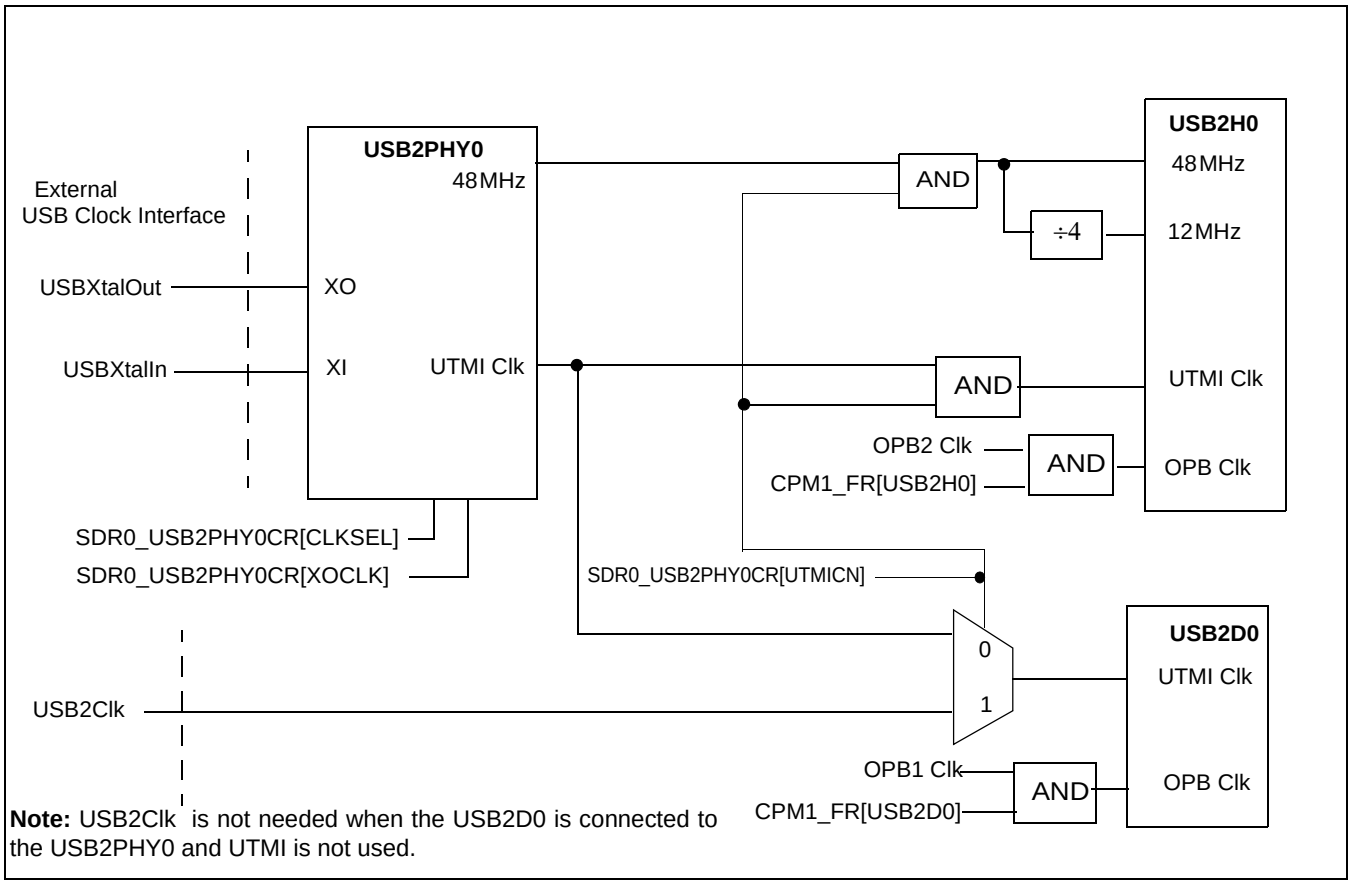
To connect USB2PHY0 to USB2D0 configure as follows:

- SDR0_USB2PHY0CR[UTMICN] = 0
- SDR0_USB2H0CR[WDINT] = 0 and SDR0_USB2PHY0CR[WDINT] = 0
- SDR0_USB2PHY0CR[DWNSTR] = 0
- SDR0_USB2PHY0CR[DVBUS] = 1

32.4 USB 2.0 Clocking

This section describes the USB clocking. The following figure illustrates the clocking architecture:

Figure 32-6. USB 2.0 Clocking



Note: USB2Clk is not needed when the USB2D0 is connected to the USB2PHY0 and UTMI is not used.

The USB2PHY0 supports the following clock sources :

- Crystal connected to the USB2XtalIn and USB2XtalOut chip inputs: The crystal must have a fundamental frequency of 48 MHz with a frequency tolerance of ±100ppm, peak jitter ±100ps, and an output differential voltage of no less than 500mV with respect to the USBXtalIn signal.
- External clock connected to the USB2XtalOut chip input: The clock must have a fundamental frequency of 48 MHz with a frequency tolerance of ±100ppm, peak jitter ±100ps, a 40/60 min/max duty cycle, and a signal swing of 3.3V.

The following table describes the settings required for each clock source.

Table 32-2. Clock Source Settings

External Clock Source	SDR0_USB2PHY0CR[XOCLK]	SDR0_USB2PHY0CR[CLKSEL]
Crystal	1	01 48 MHz
Oscillator	0	01 48 MHz

32.5 USB 2.0 POR Requirements

This section describes the power-on reset sequence required to correctly start the USB 2.0 PHY and controllers.

All the functions associated with USB are maintained under reset after deassertion of the chip reset . These functions are:

- USB2PHY0
- USB2H0
- USB2D0
- OPBA1
- OPBA2
- PLB4-to-OPB0
- PLB4-to-OPB1
- OPB-to-PLB40

To correctly start the USB functions, the user must properly deassert the reset of each of these functions as follows

- Reset of OPBA1, OPBA2, PLB4-to-OPB0, PLB4-to-OPB1, OPB-to-PLB40 cores can be deasserted immediately
- Reset of USB2PHY0 must be active at least 10µs.

  Note: The values in registers SDR0_USB2PHY0CR, SDR0_USB2D0CR and SDR0_USB2H0CR must be set while USB20PHY0 reset is active.
- After de-assertion of the USB2PHY0 reset, the controller (Device or Host depending on the setting of SDR0_USB2PHY0CR[UTMICN]) attached to the PHY can be deasserted after:
  - One millisecond if the PHY reference clock is from a crystal
  - Fifty microseconds if the PHY reference clock is from an external board clock

When the Device function is implemented with the external UTMI interface (USB2D0 not connected to USBPHY0), the USB2D0 reset can be deasserted as soon as the external clock on the chip is available.

32.6 USB 2.0 Host Interface

32.6.1 Overview

The USB 2.0 OHCI/EHCI One Port UTMI+ Host Controller with OPB Bridge Interface is based on the Synopsys® DesignWare USB 2.0 Host Subsystem-AHB core. The core implements the host controller functions defined in the Universal Serial Bus (USB), Revision 2.0, specification, the Enhanced Host Controller Interface (EHCI) specification, and the Open Host Controller Interface (OHCI) specification. The host controller supports high-speed (480Mbps), full speed (12Mbps), and low speed (1.5Mbps) transfers.

The function consists of:

- One USB EHCI host controller that handles high-speed transfers
- One USB OHCI host controller that handles full and low speed transfers
- A root hub that selects the appropriate host controller to connect to the USB port
- An AMBA AHB bus interface unit
- AHB-to-OPB bridges which provide the OPB interface

The controller supports one USB port through an on-chip IBM USB 2.0 Physical Layer Transceiver (1 port PHY) function.

The controller is fully compliant to the following specifications:

- Universal Serial Bus Specification, Revision 2.0
- Enhanced Host Controller Interface (EHCI) Specification for USB, Revision 1.0
- Open Host Controller Interface (OHCI) Specification for USB, Revision 1.0a
- *USB 2.0 Universal Transceiver Macrocell Interface (UTMI) Specification, Version 1.05*

32.6.2 Functional Description

The USB 2.0 Host function is designed with an OPB interface at the application side and a Universal Transceiver Macrocell Interface (UTMI) at the transceiver side. The function contains the Synopsys DesignWare (DW) USB 2.0 Host Subsystem-AHB with the Advanced High-Performance Bus (AHB) application interface. The Synopsys DW USB 2.0 Host Subsystem-AHB contains one OHCI Host Controller which handles full/low speed transfers (12Mbps/1.5Mbps) and one EHCI Host Controller which handles high speed transfers (480Mbps). The OPB interface is provided through the OPB Slave-to-AHB Master bridge, and through the AHB Slave-to-OPB Master bridge. The internal interface between the bridges and the Synopsys DW USB 2.0 Host Subsystem AHB is AHB. All signals traversing the clock domains (OPB, AHB, OHCI, and EHCI host controller clocks) are synchronized internally in the subsystem.

The OPB slave interface is used by the Host Controller Driver (HCD) to access the EHCI host controller capability, operational, and Synopsys specific registers and the OHCI host controller operational registers. The OPB master interface is used by the OHCI and EHCI host controllers to access the USB data structures and data buffers defined by the HCD.

The USB 2.0 Host function's connection to the USB wire is provided through a one-port UTMI PHY transceiver interface. This interface is connected to the USB 2.0 Physical Layer function (Single port USB 2.0 Physical Layer UTMI+ version 1.05).

32.6.3 Power Management

Power management for the USB function operates as described in the following sections.

32.6.3.1 USB EHCI Host Controller

The EHCI Host Controller provides power management capabilities under control of system software. The first step is to suspend the USB port if it is enabled. This halts the UTMI port PHY clocks. The next step is to shut off the host controller by setting the Run/Stop bit in the USBCMD register to a zero. The application software must stop all clocks and activities on the bus to place the EHCI Host Controller in low-power mode.

When a wake event occurs the system resumes operation and system software eventually sets the Run/Stop bit to a one and resumes the suspended ports. Software must not set the Run/Stop bit to a one until it is confirmed that the clock to the host controller is stable.

32.6.3.2 USB OHCI Host Controller

The OHCI Host Controller does not provide power management capabilities to switch the 12-MHz and 48-MHz clocks on/off for power savings.

32.6.4 USB 2.0 Host Registers

The USB 2.0 Host function contains four independent sets of registers:
- EHCI host controller Capability, Operational and Synopsys specific registers
- OHCI host controller Operational registers
- OPB Master AHB-to-OPB Bridge DCR Operational registers
- OPB Slave OPB-to-AHB Bridge DCR Operational registers

Table 32-3 lists the registers provided for the USB 2.0 Host function. The following sections define the bit-level detail for the registers.

.

Table 32-3. USB Host Registers

Register Name	Description	Offset	Access	Page
EHCI Registers				
EHCI0_HCCAPBASE	Version/Capability Length	0x0 E000 0300	R	820
EHCI0_HCSPARAMS	Structural Parameters	0x0 E000 0304	R	820
EHCI0_HCCPARAMS	Capability Parameters	0x0 E000 0308	R	822
EHCI0_USBCMD	USB Command	0x0 E000 0310	See register	823
EHCI0_USBSTS	USB Status	0x0 E000 0314	R/W	825
EHCI0_USBINTR	USB Interrupt Enable	0x0 E000 0318	R/W	827
EHCI0_FRINDEX	EHCI Frame Index	0x0 E000 031C	R/W	828
EHCI0_CRTLSEG	Control Data Structure Segment	0x0 E000 0320	R/W	829
EHCI0_PRDLISTB	Periodic Frame List Base Address	0x0 E000 0324	R/W	829
EHCI0_ASYNCLSTA	Current Asynchronous List Address	0x0 E000 0328	R/W	830
EHCI0_CFGFLAG	Configure Flag	0x0 E000 0350	R/W	830
EHCI0_PORTSC	Port Status and Control	0x0 E000 0354	R/W	831
EHCI0_INSNREG0	Programmable Microframe Base Value	0x0 E000 0390	R/W	835
EHCI0_INSNREG1	Programmable Packet Buffer Threshold	0x0 E000 0394	R/W	836
EHCI0_INSNREG2	Programmable Packet Buffer Depth	0x0 E000 0398	R/W	837
EHCI0_INSNREG3	Break Memory Transfer	0x0 E000 039C	R/W	837
EHCI0_INSNREG4	Debug Only	0x0 E000 03A0	R/W	838
OHCI Registers				
OHCI0_HCREV	Revision	0x0 E000 0400	R	838
OHCI0_HCCTRL	Control	0x0 E000 0404	R/W	839
OHCI0_HCCMDSTS	Command Status	0x0 E000 0408	See register	840
OHCI0_HCINTSTS	Interrupt Status	0x0 E000 040C	R/W	842
OHCI0_HCINTE	Interrupt Enable	0x0 E000 0410	R/W	843
OHCI0_HCINTDIS	Interrupt Disable	0x0 E000 0414	R/W	844

Table 32-3. USB Host Registers (continued)

Register Name	Description	Offset	Access	Page
OHCI0_HCHCCA	Host Controller Communications Area	0x0 E000 0418	R/W	845
OHCI0_HCPCED	Periodic Current Endpoint Descriptor	0x0 E000 041C	R/W	845
OHCI0_HCCHED	Control Head Endpoint Descriptor	0x0 E000 0420	R/W	846
OHCI0_HCCTRLCED	Control Current Endpoint Descriptor	0x0 E000 0424	R/W	846
OHCI0_HCBULKHED	Bulk Head Endpoint Descriptor	0x0 E000 0428	R/W	847
OHCI0_HCBULKCED	Bulk Current Endpoint Descriptor	0x0 E000 042C	R/W	847
OHCI0_HCDHEAD	Done Head Transfer Descriptor	0x0 E000 0430	R	848
OHCI0_HCFMINT	Frame Interval	0x0 E000 0434	R/W	849
OHCI0_HCFMREM	Frame Remaining	0x0 E000 0438	R	850
OHCI0_HCFMNUM	Frame Number	0x0 E000 043C	R	850
OHCI0_HCPRDSTRT	Periodic Start	0x0 E000 0440	R/W	851
OHCI0_HCLSTHRES	Low Speed Threahold	0x0 E000 0444	R/W	851
OHCI0_HCRHDESCA	Root Hub Descriptor A	0x0 E000 0448	See register	852
OHCI0_HCRHDESCB	Root Hub Descriptor B	0x0 E000 044C	R/W	854
OHCI0_HCRHSTS	Root Hub Status	0x0 E000 0450	See register	855
OHCI0_HCPRSTS	Root Hub Port Status	0x0 E000 0454	R/W	856
AHB-to-OPB DCRs				
USB2HMDCR0_BCNTL	Bridge Control (DCR)	0x0360	R/W	858
USB2HMDCR0_BSTAT	Bridge Status (DCR)	0x0361	R/W	859
USB2HMDCR0_LAT	Bridge OPB Latency (DCR)	0x0366	R/W	860
USB2HMDCR0_REVID	Bridge Revision ID (DCR)	0x0367	R	861
OPB-to-AHB DCRs				
USB2HSDCR0_BCNTL	Bridge Control (DCR)	0x0370	R/W	862
USB2HSDCR0_REVID	Bridge Revision ID (DCR)	0x0373	R	863

32.6.4.1 Version/Capability Length Register (EHCI0_HCCAPBASE)

This register contains the 2-byte interface version number and a 1-byte capability register length.

Offset: 0x0000

Size (bits): 32

Access: Read only

Reset: 0x0100_0010

Figure 32-7. Version/Capability Length Register (EHCI0_HCCAPBASE)

Bit	Mnemonic	Description	Comments
0:15	HCIVER	Interface version number EHCI revision number supported by this Host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.	
16:11		Reserved	
24:31	CAPLTH	Capability register length Offset to add to register base to find the beginning of the Operational Register Space	

32.6.4.2 Structural Parameters Register (EHCI0_HCSPARAMS)

This register contains the structural parameters, such as the number of downstream ports, port indicators, and so on.

Offset: 0x0004

Size (bits): 32

Access: Read only

Reset: 0x0000_1111

Figure 32-8. Structural Parameters Register (EHCI0_HCSPARAMS)

Bit	Mnemonic	Description	Comments
0:7		Reserved	
8:11	DPN	Debug Port Number Identifies which of the host controller ports is the debug port. The value is the port number (one-based) of the debug port. A non-zero value in this field indicates the presence of a debug port.	The value in this register must not be greater than EHCI0_HCSPARAMS[NPORTS].
12:14		Reserved	
15	PIND	Port Indicators Indicates whether the ports support port indicator control. When this bit is 1, the port status and control registers include a R/W field for controlling the state of the port indicator.	
16:19	NCC	Number of Companion Controllers Number of companion controllers associated with this USB 2.0 host controller. 0000 No companion host controllers. Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports. 0001 A non-zero value in this field indicates there are companion USB 1.1 host controller(s). Port-ownership hand-offs are supported. High, full- and low-speed devices are supported on the host controller root ports.	
20:23	NPCC	Number of Ports per Companion Controller Number of ports supported per companion host controller.	Used to indicate the port routing configuration to system software. The value in this field must be consistent with EHCI0_HCSPARAMS[NPORTS] and EHCI0_HCSPARAMS[NCC].
24	PRR	Port Routing Rules Indicates the method used by this implementation for mapped of ports to companion controllers. 0 First NPCC port is routed to the lowest numbered function companion host controller, the next NPCC port is routed to the next lowest function companion controller, and so on. 1 Port routing is explicitly enumerated by the first NPORTS elements of the HCSP-PORTROUTE array.	
25:26		Reserved	
27	PPC	Port Power Control Indicates whether the host controller implementation includes port power control. 0 Ports do not have port power switches 1 Ports have port power switches	The value of this field affects the functionality of the Port Power field in each port status and control register.
28:31	NPORTS	Number of Downstream Ports Specifies the number of physical downstream ports implemented on this host controller.	The value of this field determines how many port registers are addressable in the Operational Register Space. Valid values are in the range of 0x1 to 0xF. A zero in this field is undefined.

32.6.4.3 Capability Parameters Register (EHCI0_HCCPARAMS)

This register provides USB Multiple Mode control (time-base bit functionality), addressing capability.

Offset: 0x0008

Size (bits): 32

Access: Read only

Reset: 0x0000_A016

Figure 32-9. Capability Parameters Register (EHCI0_HCCPARAMS)

Bit	Mnemonic	Description	Comments
0:15		Reserved	
16:23	EECP	EHCI Extended Capabilities Pointer Indicates the existence of a capabilities list. A value of zero indicated no extended capabilities are implemented. A non-zero value indicates the offset in configuration space of the first EHCI extended capability.	Default is implementation dependent. The pointer value must be 0x40 or greater if implemented to maintain the consistency of the PCI header defined for this class of device.
24:27	IST	Isochronous Scheduling Threshold Indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit 24 is 0, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit 24 is 1, host software assumes the host controller may cache an isochronous data structure for an entire frame.	Default is implementation dependent.
28		Reserved	
29	ASPC	Asynchronous Schedule Park Capability 0 Park feature is not supported 1 Park feature for high-speed queue heads in the Asynchronous Schedule is supported.	Default is implementation dependent See EHCI0_USBCMD[ASPME] and EHCI0_USBCMD[ASPMC].
30	PFLF	Programmable Frame List Flag 0 System software must use a frame list length of 1024 elements 1 System software can specify and use a smaller frame list and configure the Host Controller by means of the EHCI0_USBCMD register Frame List Size field.	Default is implementation dependent. The frame list must always be aligned on a 4K page boundary. This ensures that the frame list is always physically contiguous.
31	64BAC	64-bit Addressing Capability Addressing range capability 0 32-bit address memory pointers 1 64-bit address memory pointers	This is not tightly coupled with the USBBASE address register mapping control. Sixty-four bit Addressing Capability indicates the host controller can generate 64-bit addresses as a master. The USBBASE register indicates the host controller only needs to decode 32-bit addresses as a slave.

32.6.4.4 USB Command Register (EHCI0_USBCMD)

This register indicates the command to be executed by the USB Host Controller. Writing to this register causes a command to be executed.

Offset: 0x0010

Size (bits): 32

Access: See field specification

Reset: 0x0008_0B00

Figure 32-10. USB Command Register (EHCI0_USBCMD)

Bit	Mnemonic	Description	Comments
0:7		Reserved	
8:15	ITC	Interrupt Threshold Control Select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. 0x00 Reserved 0x01 1 micro-frame 0x02 2 micro-frames 0x04 4 micro-frames 0x08 8 micro-frames (default, equates to 1ms) 0x10 16 micro-frames (2ms) 0x20 32 micro-frames (4ms) 0x40 64 micro-frames (8ms)	Software modifications to this bit while EHCI0_USBSTS[HCH] = 0 results in undefined behavior.
16:19		Reserved	
20	ASPME	Asynchronous Schedule Park Mode Enable 0 Disable Park Mode 1 Enable Park Mode	If EHCI0_HCCPARAMS[ASPC] = 1, ASPME defaults to 1 and is R/W. Otherwise, ASPME must be 0 and is Read only.
21		Reserved	
22:23	ASPMC	Asynchronous Schedule Park Mode Count Contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x1 to 0x3.	If EHCI0_HCCPARAMS[APMC] = 1, this field defaults to 0x3 and is R/W. Otherwise, it defaults to 0 and is Read only. Software must not write 0 to this bit when ASPME = 1, as this results in undefined behavior.
24	LHRC	Light Host Controller Reset 0 Light Host Controller Reset has completed and it is safe for host software to re-initialize the host controller 1 Light Host Controller Reset has not yet completed	This control bit is not required. If implemented, it allows the driver to reset the EHCI controller without affecting the state of the ports or the relationship to the companion host controllers. If not implemented, a read of this field always returns a zero.

Figure 32-10. USB Command Register (EHCI0_USBCMD) (continued)

Bit	Mnemonic	Description	Comments
25	IAAD	Interrupt on Async Advance Doorbell 0 Host Controller should not issue an interrupt the next time it advances the asynchronous schedule 1 Host Controller should issue an interrupt the next time it advances the asynchronous schedule	When the Host Controller has evicted all appropriate cached schedule states, it sets EHCI0_USBSTS[IAA] = 1. If EHCI0_USBSTS[IAAE] = 1 the Host Controller asserts an interrupt at the next interrupt threshold. The host controller sets this bit to 0 after it sets EHCI0_USBSTS[IAA] = 1.Software should not write a one to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results.
26	ASE	Asynchronous Schedule Enable 0 Do not process the Asynchronous Schedule 1 Process the Asynchronous Schedule	Default = 0 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
27	PSE	Periodic Schedule Enable 0 Do not process the Periodic Schedule 1 Process the Periodic Schedule	Default = 0 Use the PERIODICLISTBASE register to access the Periodic Schedule.
28:29	FLS	Frame List Size Specifies the size of the frame list. The size the frame list controls which bits in the Frame Index Register should be used for the Frame List Current index. 00 1024 elements (4096 bytes) 01 512 elements (2048 bytes) 10 256 elements (1024 bytes) 11 Reserved	Default = 00 This field is R/W only if EHCI0_HCCPARAMS[PFLF] =1.
30	HCRST	Host Controller Reset 0 Host Controller reset process is complete 1 Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on the USB is immediately terminated.	A USB reset is not driven on downstream ports. All operational registers, including port registers and port state machines are set to their initial values. Port ownership reverts to the companion host controller(s). Software must reinitialize the host controller in order to return the host controller to an operational state. Software cannot terminate the reset process early by writing 0 to this bit. Software should not set this bit to 1 when EHCI0_USBSTS[HCH] = 0. Attempting to reset an actively running Host Controller results in undefined behavior.
31	RS	Run/Stop 0 Stop. Host Controller completes the current and any actively pipelined transactions on the USB and then halts. 1 Run. Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1.	Default = 0 The Host Controller must halt within 16 micro-frames after software clears the RS bit. EHCI0_USBSTS[HCH] indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. Software must not write a one to this field unless EHCI0_USBSTS[HCH] = 1. Doing so produces undefined results.

32.6.4.5 USB Status Register (EHCI0_USBSTS)

This register indicates pending interrupts and various states of the Host Controller. The status resulting from a transaction on the USB is not indicated in this register. Software sets a bit to 0 in this register by writing a 1 to it.

Offset: 0x0014

Size (bits): 32

Access: See field specification

Reset: 0x0000_1000

Figure 32-11. USB Status Register (EHCI0_USBSTS)

Bit	Mnemonic	Description	Comments
0:15		Reserved	
16	ASS	Asynchronous Schedule Status Reports the current real status of the Asynchronous Schedule. 0 Disable status of the Asynchronous Schedule 1 Enable status of the Asynchronous Schedule	Default = 0 Read only The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software changes EHCI0_USBCMD[ASE]. When this bit and EHCI0_USBCMD[ASE] are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0).
17	PSS	Periodic Schedule Status Reports the current real status of the Periodic Schedule. 0 Disable status of the Periodic Schedule 1 Enable status of the Periodic Schedule	Default = 0 Read only The Host Controller is not required to immediately disable or enable the Periodic Schedule when software changes EHCI0_USBCMD[PSE]. When this bit and EHCI0_USBCMD[PSE] are the same value, the Periodic Schedule is either enabled (1) or disabled (0).
18	R	Reclamation Detects an empty asynchronous schedule.	Default = 0 Read only
19	HCH	Host Controller Halted 0 Run/Stop bit = 1. 1 Host Controller has stopped executing as a result of the Run/Stop bit set to 0 by either software or the Host Controller hardware (for example, internal error).	Default = 1 Read only
20:25		Reserved	
26	IAA	Interrupt on Async Advance Status of EHCI0_USBCMD[IAAD].	Default = 1 R/W Clear
27	HSE	Host System Error 0 No error during a host system access 1 Error during a host system access	When this error occurs, the Host Controller clears EHCI0_USBCMD[RS] to prevent further execution of the scheduled Transfer Descriptors (TDs).

Figure 32-11. USB Status Register (EHCI0_USBSTS) (continued)

Bit	Mnemonic	Description	Comments
28	FLR	Frame List Rollover 0 Frame List Index has not rolled over from its maximum value 1 Frame List Index has rolled over from its maximum value to 0	R/W Clear The exact value at which rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in EHCI0_USBCMD[FLS]) is 1024, the Frame Index Register rolls over every time EHCI0_FRINDEX[18] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time EHCI0_FRINDEX[17] toggles.
29	PCD	Port Change Detect The Host Controller sets this bit to 1 when: Any port for which the Port Owner bit is set to 0 changes from a 0 to 1 or a Force Port Resume bit changes from 0 to 1 as a result of a J-K transition detected on a suspended port. or As a result of the Connect Status Change being set to 1 after system software has relinquished ownership of a connected port by writing a 1 to a port's Port Owner bit.	R/W Clear This bit is allowed to be maintained in the Auxiliary power well. Alternatively, it is also acceptable that on a D3 to D0 transition of the EHCI HC device, this bit is loaded with the OR of all of the PORTSC change bits (including: Force port resume, over-current change, enable/disable change and connect status change).
30	ERRINT	USB Error Interrupt The Host Controller sets this bit 1 when completion of a USB transaction results in an error condition (for example, error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both ERRINT and USBINT bit are set.	R/W Clear
31	USBINT	USB Interrupt The Host Controller sets this bit to 1: On the completion of a USB transaction, which results in the retirement of a TD that had its IOC bit set. or When a short packet is detected.	R/W Clear

32.6.4.6 USB Interrupt Enable Register (EHCI0_USBINTR)

This register enables and disables reporting of the corresponding interrupt to the software. When a bit is set and the corresponding interrupt is active, an interrupt is generated to the host. Interrupt sources that are disabled in this register still appear in the EHCI0_USBSTS register to allow the software to poll for events.

Each interrupt enable bit description indicates whether it is dependent on the interrupt threshold mechanism.

Offset: 0x0018

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-12. USB Interrupt Enable Register (EHCI0_USBINTR)

Bit	Mnemonic	Description	Comments
0:26		Reserved	
26	IAAE	Interrupt on Async Advance Enable When this bit = 1 and EHCI0_USBSTS[IAA] = 1, the Host Controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by clearing EHCI0_USBSTS[IAA].	
27	HSEE	Host System Error Enable When this bit = 1 and EHCI0_USBSTS[HSE] = 1, the Host Controller issues an interrupt. The interrupt is acknowledged by clearing EHCI0_USBSTS[HSE].	
28	FLRE	Frame List Rollover Enable When this bit = 1 and EHCI0_USBSTS[FLR] = 1, the Host Controller issues an interrupt. The interrupt is acknowledged by clearing EHCI0_USBSTS[FLR].	
29	PCDE	Port Change Detect Enable When this bit = 1 and EHCI0_USBSTS[PCD] = 1, the Host Controller issues an interrupt. The interrupt is acknowledged by clearing EHCI0_USBSTS[PCD].	
30	USBEIE	USB Error Interrupt Enable When this bit = 1 and EHCI0_USBSTS[ERRINT] = 1, the Host Controller issues an interrupt. The interrupt is acknowledged by clearing EHCI0_USBSTS[ERRINT].	
31	USBIE	USB Interrupt Enable When this bit = 1 and EHCI0_USBSTS[USBINT] = 1, the Host Controller issues an interrupt. The interrupt is acknowledged by clearing EHCI0_USBSTS[USBINT]	

32.6.4.7 EHCI Frame Index Register (EHCI0_FRINDEX)

This register is used by the Host Controller to index into the periodic frame list. The register updates every 125 μs (once each micro-frame). Bits [N:3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in EHCI0_USBCMD[FLS].

This register must be written as a double word. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the Halted state as indicated by EHCI0_USBSTS[HCH]. A write to this register while EHCI0_USBSTS[RS] = 1 produces undefined results. Writes to this register also affect the SOF value.

The SOF frame number value for the bus SOF token is derived or alternatively managed from this register. The value of FRINDEX must be 125 μs (1 micro-frame) ahead of the SOF token value. The SOF value can be implemented as an 11-bit shadow register. For this discussion, this shadow register is 11 bits and is named SOFV. SOFV updates every 8 micro-frames. (1 ms). An example implementation to achieve this behavior is to increment SOFV each time the FRINDEX[2:0] increments from 0 to 1.

Software must use the value of FRINDEX to derive the current micro-frame number, both for high-speed isochronous scheduling purposes and to provide the get micro-frame number function required for client drivers. Therefore, the value of FRINDEX and the value of SOFV must be kept consistent if the chip is reset or software writes to FRINDEX. Writes to FRINDEX must also write-through FRINDEX[13:3] to SOFV[10:0]. In order to keep the update as simple as possible, software should never write a FRINDEX value where the three least significant bits are 0b111 or 0b000.

Offset: 0x001C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-13. EHCI Frame Index Register (EHCI0_FRINDEX)

Bit	Mnemonic	Description	Comments
0:17		Reserved	
18:31	FI	Frame Index The value in this register increments at the end of each time frame (for example, a micro-frame). Bits [N:3] are used for the Frame List current index. This means that each location of the frame list is accessed eight times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register: USBCMD[FLS] Number Elements N 00 1024 12 01 512 11 10 256 10 11 Reserved	

32.6.4.8 Control Data Structure Segment Register (EHCI0_CRTLSEG)

This register corresponds to the most significant address bits [63:32] for all EHCI data structures. If EHCI0_HCCPARAMS[64BAC] = 0, this register is not used. Software cannot write to it, and a read from this register returns zeros.

If the EHCI0_HCCPARAMS[64BAC] = 1, this register is used with the link pointers to construct 64-bit addresses to EHCI control data structures. This register is concatenated with the link pointer from either the PERIODICLISTBASE, ASYNCLISTADDR, or any control data structure link field to construct a 64-bit address.

This register allows software to locate all control data structures within the same 4GB memory segment.

Offset: 0x0020

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-14. Control Data Structure Segment Register (EHCI0_CRTLSEG)

Bit	Mnemonic	Description	Comments
0:31	4GCDA	4GB Control Data Structure Segment Address	

32.6.4.9 Periodic Frame List Base Address Register (EHCI0_PRDLISTB)

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. If the Host Controller is in 64-bit mode (EHCI0_HCCPARAMS[64BAC] = 1) , the most significant 32 bits of every control data structure address comes from EHCI0_CRTLSEG. System software loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-KB aligned. The contents of this register are combined with EHCI0_FRINDEX to enable the Host Controller to step through the Periodic Frame List in sequence.

Offset: 0x0024

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-15. Periodic Frame List Base Address Register (EHCI0_PRDLISTB)

Bit	Mnemonic	Description	Comments
0:19	LOW	Base Address Corresponds to memory address signals 0:19, respectively.	
20:31		Reserved	

32.6.4.10 Current Asynchronous List Address Register (EHCI0_ASYNCLSTA)

This register contains the address of the next asynchronous queue head to be executed. If the Host Controller is in 64-bit mode (EHCI0_HCCPARAMS[64BAC] = 1), the most significant 32 bits of every control data structure address comes from EHCI0_CRTLSEG. The memory structure referenced by this physical memory pointer is assumed to be 32-byte (cache line) aligned.

Offset: 0x0028

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-16. Current Asynchronous List Address Register (EHCI0_ASYNCLSTA)

Bit	Mnemonic	Description	Comments
0:26	LPL	Link Pointer Low Corresponds to memory address signals 0:26, respectively.	This field may only reference a Queue Head (QH).
27:31		Reserved	

32.6.4.11 Configure Flag Register (EHCI0_CFGFLAG)

This register is in the auxiliary power well. It is only reset by hardware when the auxiliary power is initially applied or in response to a Host Controller reset.

Offset: 0x0050

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-17. Configure Flag Register (EHCI0_CFGFLAG)

Bit	Mnemonic	Description	Comments
0:30		Reserved	
31	CF	Configure Flag Controls the default port-routing control logic. 0 Port routing control logic default-routes each port to an implementation dependent classic Host Controller. 1 Port routing control logic default-routes all ports to this Host Controller.	Host software sets this bit as the last action in its process of configuring the Host Controller.

32.6.4.12 Port Status and Control Register (EHCI0_PORTSC)

A Host Controller must implement one or more port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in EHCI0_HCCPARAMS. Software uses this information as an input parameter to determine how many ports need to be serviced. All ports have the structure defined below.

This register is in the auxiliary power well. It is only reset by hardware when the auxiliary power is initially applied or in response to a host controller reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port has port power control, software cannot change the state of the port until after it applies power to the port by setting port power to a 1. Software must not attempt to change the state of the port until after power is stable on the port. The host is required to have power stable to the port within 20ms of the 0 to 1 transition.

Notes:

1. When a device is attached, the port state changes to the connected state and system software processes this as with any status change notification.
2. If a port is being used as the Debug Port, then the port may report device connected and enabled when the Configured Flag is 0.

Offset: 0x0054

Size (bits): 32

Access: See field specification

Reset: 0x0000_2000

Table 32-4. Port Status and Control Register (EHCI0_PORTSC)

Bit	Mnemonic	Description	Comments
0:8		Reserved	
9	WKOCE	Wake on Over-current Enable Writing this bit to a 0 Port Power is 0 1 Enables the port for sensitivity to over-current conditions as wake-up events	R/W
10	WKDCTE	Wake on Disconnect Enable 0 Port Power is 0 1 Enables the port for sensitivity to device disconnects as wake-up events	R/W
11	WKCTE	Wake on Connect Enable 0 Port Power is 0 1 Enables the port for sensitivity to device connects as wake-up events	R/W

Table 32-4. Port Status and Control Register (EHCI0_PORTSC) (continued)

Bit	Mnemonic	Description	Comments
12:15	PTC	Port Test Control 0000 Port is NOT operating in a test mode 0001 Test J state 0010 Test K state 0011 Test SE0_NAK 0100 Test packet 0101 Test FORCE_ENABLE 0110 ... 1111 Reserved	R/W
16:17	PIC	Port Indicator Control If EHCI0_HCCPARAMS[PIND] is 1, then: 00 Port Power is 0 or port indicators are off 01 Amber 10 Green 11 Undefined	R/W
18	PO	Port Owner 0 EHCI0_CFGFLAG[CF] made a 0 to 1 transition 1 EHCI0_CFGFLAG[CF] is 0	R/W System software uses this field to release ownership of the port to a selected Host Controller (in the event that the attached device is not a high-speed device). Software writes 1 to this bit when the attached device is not a high-speed device.
19	PP	Port Power The function of this bit depends on the value of EHCI0_HCCPARAMS[PPC] 0 Port power control switch is off 1 Port power control switch is on	R/W This bit is Read only if the Host Controller does not have port power control switches. Each port is hard-wired to power. When power is not available on a port (that is, PP = 0), the port is nonfunctional and does not report attaches, detaches, and so on. When an over-current condition is detected on a powered port and PPC is 1, the PP bit in each affected port can be changed by the Host Controller from a 1 to 0 (removes power from the port).
20:21	LS	Line Status These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence.. USB State Interpretation 00 SE0 Not low-speed device, perform EHCI reset 01 J-state Not low-speed device, perform EHCI reset 10 K-state Low-speed device, release ownership of port 11 Undefined Not Low-speed device, perform EHCI reset.	Read only Undefined if EHCI0_PORTSC[PP] = 0. This field is valid only when the port enable bit is 0 and the current connect status bit is 1.
22		Reserved	

Table 32-4. Port Status and Control Register (EHCI0_PORTSC) (continued)

Bit	Mnemonic	Description	Comments
23	PR	Port Reset 0 Port is not in reset 1 Port is in reset	R/W When software changes this bit from 0 to1, the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes 0 to this bit to terminate the bus reset sequence. Software must keep this bit at 1 long enough to ensure the reset sequence completes. **Note:** When software writes 1 to this bit, it must also write 0 to the Port Enable bit. When software writes 0 to this bit there might be a delay before the bit status changes to a 0. The bit status will not read as a 0 until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (that is, set the Port Enable bit to 1). A Host Controller must terminate the reset and stabilize the state of the port within 2 ms of software changing this bit from 1 to 0. For example, if the port detects that the attached device is high-speed during reset, the Host Controller must have the port in the enabled state 2ms of software setting this bit to 0. EHCI0_USBSTS[HCH] should be 0 before software attempts to use this bit. The Host Controller may hold Port Reset asserted to 1 when EHCI0_USBSTS[HCH] = 1. This field is 0 if Port Power is 0.
24	S	Suspend 0 Port not in suspended state 1 Port in suspended state	R/W The Port Enabled and Suspend bits of this register define the port states as follows: Bits Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. A write of zero to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a zero when: • Software changes the Force Port Resume bit from 1 to 0. • Software changes the Port Reset bit from 0 to1. If Host software sets this bit to a one when the port is not enabled (that is, Port Enabled = 0) the results are undefined. This field is 0 if Port Power is 0.

Table 32-4. Port Status and Control Register (EHCI0_PORTSC) (continued)

Bit	Mnemonic	Description	Comments
25	FPR	Force Port Resume 0 No resume (K-state) detected/driven on port 1 Resume detected/driven on port	R/W The functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Enabled bits are a one) and software changes this bit to 1, the effects on the bus are undefined. Software sets this bit to 1 to drive resume signaling. The Host Controller sets this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit changes to 1 because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is set to 1. If software sets this bit to 1, the hOst Controller must not set the Port Change Detect bit. When the EHCI controller owns the port, the resume sequence follows the sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. Software must appropriately time the Resume and set this bit to 0 when the appropriate amount of time has elapsed. Changing the bit from 0 to 1 causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit remains 1 until the port switches to the high-speed idle. The host controller must complete this transition within 2ms of software setting this bit to 0. This field is zero if Port Power is zero.
26	OCC	Over Current Change 0 No change to over current active 1 Change to over current active occurred.	Read with Clear Software clears this bit by writing 1.
27	OCA	Over Current Active 0 No over current condition on this port 1 Over current condition exists on this port	Read only This bit automatically clears when the over current condition is removed.
28	PEDC	Port Enable/Disable Change 0 No change 1 Port enabled/disabled status has changed	Read with Clear For the root hub, this bit is 1 only when a port is disabled due to the appropriate conditions existing at the EOF2 point. Software clears this bit by writing 1. This field is 0 if Port Power is 0.
29	PED	Port Enable/Disable 0 Disable 1 Enable	Ports can only be enabled by the Host Controller as a part of the reset and enable. Software cannot enable a port by writing 1 to this field. The host controller sets this bit to 1 only when the reset sequence determines that the attached device is a high-speed device. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. The bit status does not change until the port state actually changes. There might be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled downstream propagation of data is blocked on this port, except for reset. This field is zero if Port Power is zero.

Table 32-4. Port Status and Control Register (EHCI0_PORTSC) (continued)			
Bit	**Mnemonic**	**Description**	**Comments**
30	CSC	Connect Status Change Indicates a change has occurred in the port Current Connect Status. 0 No change 1 Change in Current Connect Status	Default = 0. The Host Controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, and hub is setting an already-set bit (that is., the bit remains set). Software clears this bit by writing 1 to it. This field is zero if Port Power is zero.
31	CCS	Current Connect Status 0 No device is present 1 Device is present on port	Default = 0. This value reflects the current state of the port, and might not correspond directly to the event that caused EHCI0_PORTSC[CSC] to be set. This field is zero if Port Power is zero.

32.6.4.13 Programmable Microframe Base Value Register (EHCI0_INSNREG0)

This register allows the user to vary the 125 µs microframe time for simulation purposes. Each decimal value change in the 1 Microframe Counter Value field corresponds to 1 UTMI PHY interface clock cycle.

Offset: 0x0090

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-18. Programmable Microframe Base Value Register (EHCI0_INSNREG0)			
Bit	**Mnemonic**	**Description**	**Comments**
0:17		Reserved	
18:30	1MFCV	1 Microframe Counter Value	Bits 19:30 are used when the UTMI PHY interface clock is 30MHz. Bits 18:30 are used when the UTMI PHY interface clock is 60MHz.
31	EMFCV	Enable Microframe Counter Value 0 Host uses the following default values: 7436 decimal for a 60 MHz UTMI PHY interface 3718 decimal for a 30MHz UTMI PHY interface. 1 Host uses the value specified in 1MFCV.	

32.6.4.14 Programmable Packet Buffer Threshold Register (EHCI0_INSNREG1)

The OUT threshold value represents the quantity of data read from system memory into the packet buffer that, when reached, the host controller starts the USB data transfer to the USB device. The IN threshold value represents the quantity of data received from a USB device into the packet buffer that, when reached, the host controller starts the transfer of the data to system memory. For systems with slow system memory access, set the threshold levels to the maximum value (packet buffer depth) to avoid USB data overruns and underruns.

For both the IN and OUT threshold values, each decimal value change in the threshold value corresponds to one DWORD (4 byte) quantity of data.

Offset: 0x0094

Size (bits): 32

Access: R/W

Reset: 0x0040_0040

Figure 32-19. Programmable Packet Buffer Threshold Register (EHCI0_INSNREG1)

Bit	Mnemonic	Description	Comments
0:16	PPBOTV	Programmable Packet Buffer OUT Threshold Value	Units equal number of DWORDs (4 bytes). The maximum value this field may be set to is 0x0100 (256 decimal x 4 = 1024 bytes, the physical packet buffer depth). Do not exceed this value when setting this field. Default value = 0x0040 (64 decimal x 4 = 256 bytes)
16:31	PPBITV	Programmable Packet Buffer IN Threshold Value	Units equal number of DWORDs (4 bytes). The maximum value this field may be set to is 0x0100 (256 decimal x 4 = 1024 bytes, the physical packet buffer depth). Do not exceed this value when setting this field. Default value = 0x0040 (64 decimal x 4 = 256 bytes)

32.6.4.15 Programmable Packet Buffer Depth Register (EHCI0_INSNREG2)

The size of the packet buffer implemented in the Host Controller is 1024 bytes. As a performance evaluation tool, this register to allow the user to adjust the size of the packet buffer smaller to aid in determining if the packet buffer size can be decreased to save area in future hardware implementations.

Each decimal value change in the Programmable Packet Buffer Depth corresponds to one DWORD (4 byte) quantity of data.

Offset: 0x0098

Size (bits): 32

Access: R/W

Reset: 0x0000_0100

Figure 32-20. Programmable Packet Buffer Depth Register (EHCI0_INSNREG2)

Bit	Mnemonic	Description	Comments
0:19		Reserved	
20:31	PPBDV	Programmable Packet Buffer Depth Value	Units equal number of DWORDs (4 bytes). The maximum value this field may be set to is 0x0100 (256 decimal x 4 = 1024 bytes, the physical packet buffer depth). Do not exceed this value when setting this field.

32.6.4.16 Break Memory Transfer Register (EHCI0_INSNREG3)

When Break Memory Transfer is enabled during packet buffer data transfers to or from system memory, the host OPB master locks the bus until the corresponding threshhold value in EHCI0_INSNREG1 has been reached. At this point, the OPB Master gives up control of the bus and rearbitrates for it to allow other OPB masters access. When Break Memory Transfer is disabled, the host OPB master locks the bus until the all of the data in the packet buffer has been transferred to or from system memory.

Offset: 0x009C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-21. Break Memory Transfer Register (EHCI0_INSNREG3)

Bit	Mnemonic	Description	Comments
0:30		Reserved	
31	BMT	Break Memory Transfer 0 Disable break memory transfer 1 Enable break memory transfer	

32.6.4.17 Debug Only Register (EHCI0_INSNREG4)

The ESDPET field scales down the time of specific events within the USB Chirp process for debug purposes.

The EWHCCP and EWHCSP fields allows the user to modify the normally read-only registers EHCI0_HCCPARAMS and EHCI0_HCSPARAMS to accept OPB write transfers.

Offset: 0x00A0

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-22. Debug Only Register (EHCI0_INSNREG4)

Bit	Mnemonic	Description	Comments
0:28		Reserved	
29	ESDPET	Enable Scale Down Port Enumeration Time 0 Use defined time 1 Enable scaled down time	
30	EWHCCP	Enable Writable EHCI0_HCCPARAMS register 0 Read Only 1 Writable	
31	EWHCSP	Enable Writable EHCI0_HCSPARAMS register 0 Read Only 1 Writable	

32.6.4.18 Revision Register (OHCI0_HCREV)

This register contiains the revision number of the OHCI specification supported by this Host Controller.

Offset: 0x0000

Size (bits): 32

Access: Read only

Reset: 0x0000_0110

Figure 32-23. Revision Register (OHCI0_HCREV)

Bit	Mnemonic	Description	Comments
0:22		Reserved	
23	L	Legacy	
24:31	REV	OHCI revision number BCD representation of the version of the HCI specification that is implemented by this Host Controller. For example, a value of 0x11 corresponds to version 1.1.	All of the Host Controller implementations that are compliant with the above specification will have a value of 0x10.

32.6.4.19 Control Register (OHCI0_HCCTRL)

This register defines the operating modes for the Host Controller. With the exception of HCFS and RWC, the fields in this register are modified only by the Host Controller driver.

Offset: 0x0004

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-24. Control Register (OHCI0_HCCTRL)

Bit	Mnemonic	Description	Comments
0:20		Reserved	
21	RWE	Remote Wake up Enable Used by the Host Controller driver to enable or disable the remote wake up feature upon the detection of upstream resume signaling. 0 Disable remote wake up 1 Enable remote wake up	When this bit is set and OHCI0_HCCMDSTS[RD] is set, a remote wakeup is signaled to the Host system. Setting this bit has no impact on the generation of hardware interrupts.
22	RWC	Remote Wakeup Connected Indicates whether the Host Controller supports remote wakeup signaling. 0 No remote wakeup signaling 1 Remote wakeup signaling supported	If remote wakeup is supported and used by the system it is the responsibility of system firmware to set this bit during POST. The Host Controller clears the bit on a hardware reset but does not alter it on a software reset.
23	IR	Interrupt Routing Determines the routing of interrupts generated by events registered in OHCI0_HCCMDSTS. 0 Interrupts are routed to the normal host bus interrupt mechanism 1 Interrupts are routed to the System Management Interrupt	The Host Controller driver clears this bit on a hardware reset, but it does not alter this bit on a software reset. the Host Controller driver uses this bit as a tag to indicate the ownership of the Host Controller.
24:25	HCFS	Host Controller Functional State 00 USBRESET 01 USBRESUME 10 USBOPERATIONAL 11 USBSUSPEND	A transition to USBOPERATIONAL from another state causes SOF generation to begin 1ms later. The Host Controller driver can determine whether the Host Controller has begun sending SOFs by reading OHCI0_HCCMDSTS[SF]. This field can be changed by the Host Controller only when in the USBSUSPEND state. The Host Controller can move from the USBSUSPEND state to the USBRESUME state after detecting resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the root hub and asserts subsequent reset signaling to downstream ports.
26	BLE	Bulk List Enable 0 Disable processing of the Bulk list after the next SOF. 1 Enable processing of the Bulk list in the next Frame	The Host Controller checks this bit whenever it processes the list. When disabled, the Host Controller driver can modify the list. If OHCI0_HCBULKCED is pointing to an endpoint descriptor (ED) to be removed, the Host Controller driver must advance the pointer by updating OHCI0_HCBULKCED before re-enabling processing of the list.

Figure 32-24. Control Register (OHCI0_HCCTRL) (continued)

Bit	Mnemonic	Description	Comments
27	CLE	Control List Enable 0 Disable processing of the Control list after the next SOF 1 Enable processing of the Control list in the next Frame	The Host Controller must check this bit whenever it processes the list. When disabled, the Host Controller driver can modify the list. If OHCI0_HCCTRLCED is pointing to an endpoint descriptor to be removed, the Host Controller driver must advance the pointer by updating OHCI0_HCCTRLCED before re-enabling processing of the list.
28	IE	Isochronous Enable Used by the Host Controller driver to enable/ disable processing of isochronous EDs. 0 Disable processing of the periodic list (which now contains only isochronous EDs) and begin processing the Bulk/Control lists 1 Enabled processing the EDs	While processing the periodic list in a Frame, the Host Controller checks the status of this bit when it finds an Isochronous ED (F=1). Setting this bit itakes effect in the next Frame, not the current Frame.
29	PLE	Periodic List Enable 0 Disable processing of the Periodic List after the next SOF 1 Enable processing of the Periodic List in the next Frame	The Host Controller must check this bit before it starts processing the list.
30:31	CBSR	Control Bulk Service Ratio Specifies the service ratio (number of Control EDs over Bulk EDs served) between Control and Bulk EDs. 00 1:1 01 2:1 10 3:1 11 4:1	Before processing any of the nonperiodic lists, the Host Controller must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, the Host Controller driver is responsible for restoring this value.

32.6.4.20 Command Status Register (OHCI0_HCCMDSTS)

This register is used by the Host Controller to receive commands issued by the Host Controller driver, as well as reflecting the current status of the Host Controller. To the Host Controller driver, it appears to be a "write to set" register. The Host Controller must ensure that bits written as 1 become set in the register while bits written as 0 remain unchanged in the register. The Host Controller driver can issue multiple distinct commands to the Host Controller without concern for corrupting previously issued commands. The Host Controller Driver has normal Read access to all bits.

The SOC field indicates the number of frames with which the Host Controller has detected the scheduling overrun error. This occurs when the Periodic list does not complete before EOF. When a scheduling overrun error is detected, the Host Controller increments the counter and sets OHCI0_HCINTSTS[SO].

Offset: 0x0008

Size (bits): 32

Access: See field specification

Reset: 0x0000_0000

Figure 32-25. Command Status Register (OHCI0_HCCMDSTS)

Bit	Mnemonic	Description	Comments
0:13		Reserved	
14:15	SOC	Scheduling Overrun Count Incremented on each scheduling overrun error. It is initialized to 0b00 and wraps around at 0b11.	Read only This is incremented when a scheduling overrun is detected even if OHCI0_HCCMDSTS[SO] is already set. It is used by the Host Controller driver to monitor any persistent scheduling problems.
16:27		Reserved	
28	OCR	Ownership Change Request 0 Do not request a change of control of the Host Controller 1 Request a change of control of the Host Controller	R/W When set the Host Controller sets OHCI0_HCCMDSTS[OC]. After the changeover, this bit is cleared and remains so until the next request from Host Controller driver.
29	BLF	Bulk List Filled Used to indicate whether there are any TDs on the Bulk list. 0 Do not process the Bulk list 1 Host Controller driver added a TD to an ED in the Bulk list	R/W When the Host Controller begins processing the head of the Bulk list, it checks BLF. As long as BLF is 0, the Host Controller does not start . If BLF is 1, the Host Controller starts processing the Bulk list and sets BLF to 0. If the Host Controller finds a TD on the list, the Host Controller sets BLF to 1 causing the Bulk list processing to continue. If no TD is found on the Bulk list, and if the Host Controller driver does not set BLF, then BLF will still be 0 when the Host Controller completes processing the Bulk list and Bulk list processing will stop.
30	CLF	Control List Filled Used to indicate whether there are any TDs on the Control list. 0 Do not process the Control list 1 Host Controller driver added a TD to an ED in	R/W When the Host Controller begins to process the head of the Control list, it checks CLF. As long as CLF is 0, the Host Controller does not start processing the Control list. If CF is 1, the Host Controller starts processing the Control list and sets CLF to 0. If the Host Controller finds a TD on the list, then HC sets CLF to 1 causing the Control list processing to continue. If no TD is found on the Control list, and if the the Host Controller driver does not set CLF, then CLF will still be 0 when the Host Controller completes processing the Control list and Control list processing will stop.
31	HCR	Host Controller Reset 0 Reset of the Host Controller complete 1 Initiate a software reset of the Host Controller	R/W Regardless of the functional state of the Host Controller, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise (for example, OHCI0_HCCTRL[IR]), and no Host bus accesses are allowed. The reset operation must be completed within 10ms. This bit, when set, should not cause a reset to the root hub and no subsequent reset signaling should be asserted to its downstream ports.

32.6.4.21 Interrupt Status Register (OHCI0_HCINTSTS)

This register provides status on various events that cause hardware interrupts. When an event occurs, the Host Controller sets the corresponding bit in this register. When a bit becomes set, a hardware interrupt is generated if OHCI0_HCINTE[MIE] is set. The Host Controller driver can clear specific bits in this register by writing 1 to bit positions to be cleared. The Host Controller driver cannot set any of these bits. The Host Controller never clears these bits.

Offset: 0x000C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-26. Interrupt Status Register (OHCI0_HCINTSTS)

Bit	Mnemonic	Description	Comments
0		Reserved	
1	OC	Ownership Change Set by the Host Controller when the Host Controller driver sets OHCI0_HCCMDSTS[OCR].	This event, when unmasked, always generates a System Management Interrupt (SMI) immediately. This bit is tied to 0 when the SMI pin is not implemented.
2:24		Reserved	
25	RHSC	Root Hub Status Change Set when the content of OHCI0_HCRHSTS or OHCI0_HCPRSTS has changed.	
26	FNO	Frame Number Overflow Set when the msb of OHCI0_HCFMNUM[FN] changes value, from 0 to 1 or from 1 to 0, and after the Frame Number value has been updated.	
27	UE	Unrecoverable Error 0 Host Controller has been reset 1 Host Controller detected a system error not related to USB	The Host Controller should not do any processing or signaling before the system error has been corrected.
28	RD	Resume Detected 0 USBRESUME state is set 1 Device on the USB is asserting resume signaling	A transition from no resume signaling to resume signaling causes this bit to be set.
29	SF	Start of Frame Set by the Host Controller at each SOF and after the update of the Frame Number.	The Host Controller generates a SOF token at this time.
30	WDH	Write Back Done Head 0 OHCI0_HCDHEAD[DH] has been saved by the Host Controller 1 Done Head value is written to OHCI0_HCDHEAD[DH]	Further updates of the Done Head value do not occur until this bit is cleared.
31	SO	Scheduling Overrun Set when the USB schedule for the current Frame overruns and after the update of the Frame Number.	A scheduling overrun also causes OHCI0_HCCMDSTS[SOC] to be incremented.

32.6.4.22 Interrupt Enable Register (OHCI0_HCINTE)

Each enable bit in this register corresponds to an associated interrupt bit in OHCI0_HCINTSTS. This register is used to control which events generate a hardware interrupt. When a bit is set in OHCI0_HCINTSTS and the corresponding bit in this register is set and MIE is set, a hardware interrupt is requested on the host bus.

Writing 1 to a bit in this register sets the corresponding bit. Writing 0 to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Offset: 0x0010

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-27. Interrupt Enable Register (OHCI0_HCINTE)

Bit	Mnemonic	Description	Comments
0	MIE	Master Interrupt Enable 0 Ignored 1 Enables interrupt generation due to events specified in the other bits of this register	
1	OC	Ownership Change interrupt enable 0 Ignored 1 Enable interrupt	
2:24		Reserved	
25	RHSC	Root Hub Status Change interrupt enable 0 Ignored 1 Enable interrupt	
26	FNO	Frame Number Overflow interrupt enable 0 Ignored 1 Enable interrupt	
27	UE	Unrecoverable Error interrupt enable 0 Ignored 1 Enable interrupt	
28	RD	Resume Detected interrupt enable 0 Ignored 1 Enable interrupt	
29	SF	Start of Frame interrupt enable 0 Ignored 1 Enable interrupt	
30	WDH	Write Back Done Head interrupt enable 0 Ignored 1 Enable interrupt	
31	SO	Scheduling Overrun interrupt enable 0 Ignored 1 Enable interrupt	

32.6.4.23 Interrupt Disable Register (OHCI0_HCINTDIS)

Each disable bit in this register corresponds to an associated interrupt bit in OHCI0_HCINTSTS. This register is coupled with OHCI0_HCINTE. This register is used to control which events generate a hardware interrupt. When a bit is set in OHCI0_HCINTSTS and the corresponding bit in this register is set and MIE is set, a hardware interrupt is requested on the host bus.

Writing 1 to a bit in this register clears the corresponding bit. Writing 0 to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Offset: 0x0014

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-28. Interrupt Disable Register (OHCI0_HCINTDIS)

Bit	Mnemonic	Description	Comments
0	MIE	Master Interrupt Disable 0 Ignored 1 Disables interrupt generation due to events specified in the other bits of this register	This field is set after a hardware or software reset.
1	OC	Ownership Change interrupt disable 0 Ignored 1 Disable interrupt	
2:24		Reserved	
25	RHSC	Root Hub Status Change interrupt disable 0 Ignored 1 Disable interrupt	
26	FNO	Frame Number Overflow interrupt disable 0 Ignored 1 Disable interrupt	
27	UE	Unrecoverable Error interrupt disable 0 Ignored 1 Disable interrupt	
28	RD	Resume Detected interrupt disable 0 Ignored 1 Disable interrupt	
29	SF	Start of Frame interrupt disable 0 Ignored 1 Disable interrupt	
30	WDH	Write Back Done Head interrupt disable 0 Ignored 1 Disable interrupt	
31	SO	Scheduling Overrun interrupt disable 0 Ignored 1 Disable interrupt	

32.6.4.24 Host Controller Communications Area Register (OHCI0_HCHCCA)

This register contains the physical address of the Host Controller Communication Area (HCCA). The Host Controller driver determines the alignment restrictions by writing all 1s to this register and then reading the contents. The alignment is evaluated by examining the number of zeroes in the lower order bits. The minimum alignment is 256 bytes. Therefore, bits 0 through 7 must always return 0 when read. This area is used to hold the control structures and the Interrupt table that are accessed by both the Host Controller and the Host Controller driver.

Offset: 0x0018

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-29. Host Controller Communications Area Register (OHCI0_HCHCCA)

Bit	Mnemonic	Description	Comments
0:23	HCCA	Base address of the HCCA.	
24:31		Reserved	

32.6.4.25 Periodic Current Endpoint Descriptor Register (OHCI0_HCPCED)

This register contains the physical address of the current Isochronous or Interrupt Endpoint Descriptor.

Offset: 0x001C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-30. Periodic Current Endpoint Descriptor Register (OHCI0_HCPCED)

Bit	Mnemonic	Description	Comments
0:27	PCED	Period Current ED Pointer to the head of one of the periodic lists which will be processed in the current Frame.	The content of this register is updated by the Host Controller after a periodic ED has been processed. The Host Controller driver can read the content to determine which ED is being processed at the time of reading.
28:31		Reserved	

32.6.4.26 Control Head Endpoint Descriptor Register (OHCI0_HCCHED)

This register contains the physical address of the first Endpoint Descriptor of the Control list.

Offset: 0x0020

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-31. Control Head Endpoint Descriptor Register (OHCI0_HCCHED)

Bit	Mnemonic	Description	Comments
0:27	CHED	Control Head ED The Host Controller processes the Control list starting with the CHED pointer.	The content is loaded from HCCA during the initialization of the Host Controller .
28:31		Reserved	

32.6.4.27 Control Current Endpoint Descriptor Register (OHCI0_HCCTRLCED)

This register contains the physical address of the current Endpoint Descriptor of the Control list.

Offset: 0x0024

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-32. Control Current Endpoint Descriptor Register (OHCI0_HCCTRLCED)

Bit	Mnemonic	Description	Comments
0:27	CCED	Control Current ED Initially, this is set to zero to indicate the end of the Control list and no processing occurs. Otherwise, the Host Controller reads the instantaneous value of this register. This pointer is advanced to the next ED after serving the present one. The Host Controller continues processing the list from where it left off in the last Frame. When it reaches the end of the Control list, it checks OHCI0_HCCMDSTS[CLF]. If set, it copies the content of OHCI0_HCCHED to OHCI0_HCCTRLCED and clears OHCI0_HCCMDSTS[CLF].	The Host Controller driver can modify this register only when OHCI0_HCCTRL[CLE] = 0.
28:31		Reserved	

32.6.4.28 Bulk Head Endpoint Descriptor Register (OHCI0_HCBULKHED)

This register contains the physical address of the first Endpoint Descriptor of the Bulk list.

Offset: 0x0028

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-33. Bulk Head Endpoint Descriptor Register (OHCI0_HCBULKHED)

Bit	Mnemonic	Description	Comments
0:27	BHED	Bulk Head ED The Host Controller processes the Bulk list starting with the BHED pointer.	The content is loaded from HCCA during the initialization of the Host Controller.
28:31		Reserved	

32.6.4.29 Bulk Current Endpoint Descriptor Register (OHCI0_HCBULKCED)

This register contains the physical address of the current endpoint of the Bulk list. As the Bulk list is processed in a round-robin fashion, the endpoints are ordered according to their insertion to the list.

Offset: 0x002C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-34. Bulk Current Endpoint Descriptor Register (OHCI0_HCBULKCED)

Bit	Mnemonic	Description	Comments
0:27	BCED	Bulk Current ED Initially, this is set to zero to indicate the end of the Bulk list and no processing occurs. Otherwise, the Host Controller reads the instantaneous value of this register. This pointer is advanced to the next ED after serving the present one. The Host Controller continues processing the list from where it left off in the last Frame. When it reaches the end of the Bulk list, it checks OHCI0_HCCMDSTS[BLF]. If set, it copies the content of OHCI0_HCBULKHED to OHCI0_HCBULKCED and clears OHCI0_HCCMDSTS[BLF].	The Host Controller driver can modify this register only when OHCI0_HCCTRL[BLE] = 0.
28:31		Reserved	

32.6.4.30 Done Head Transfer Descriptor Register (OHCI0_HCDHEAD)

This register contains the physical address of the last completed Transfer Descriptor (TD) that was added to the Done queue. In normal operation, the Host Controller driver should not need to read this register as its content is periodically written to the HCCA.

Offset: 0x0030

Size (bits): 32

Access: Read only

Reset: 0x0000_0000

Figure 32-35. Done Head Transfer Descriptor Register (OHCI0_HCDHEAD)

Bit	Mnemonic	Description	Comments
0:27	DH	Done Head When a TD is completed, the Host Controller writes the content of this register to the NextTD field of the TD. The Host Controller then overwrites the content of this register with the address of this TD.	This is set to 0 whenever the Host Controller writes the content of this register to HCCA. It also sets OHCI0_HCINTSTS[WDH].
28:31		Reserved	

32.6.4.31 Frame Interval Register (OHCI0_HCFMINT)

This register contains a 14-bit value which indicates the bit time interval in a Frame, (that is, the data between two consecutive SOFs), and a 15-bit value indicating the Full Speed maximum packet size that the Host Controller can transmit or receive without causing scheduling overrun.

The Host Controller Driver can carry out minor adjustments on the Frame Interval by writing a new value over the present one at each SOF. This provides the programmability necessary for the Host Controller to synchronize with an external clocking resource and to adjust for any unknown local clock offset.

Offset: 0x0034

Size (bits): 32

Access: R/W

Reset: 0x0000_2EDF

Figure 32-36. Frame Interval Register (OHCI0_HCFMINT)

Bit	Mnemonic	Description	Comments
0	FIT	Frame Interval Toggle Toggles whenever a new value is loaded in Frame Interval.	
1:15	FSMPS	FS Largest Data Packet Specifies the value loaded into the Largest Data Packet Counter at the beginning of each frame.	The counter value represents the largest number of data in bits which can be sent or received by the Host Controller in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the the Host Controller.
16:17		Reserved	
18:31	FI	Frame Interval Specifies the interval between two consecutive SOFs in bit times. The nominal value is set to 11,999.	The Host Controller driver should store the current value of this field before resetting the Host Controller. Setting OHCI0_HCCMDSTS[HCR] causes the Host Controller to reset this field to its nominal value. The Host Controller driver can choose to restore the stored value upon the completion of the reset sequence.

32.6.4.32 Frame Remaining Register (OHCI0_HCFMREM)

This register is a 14-bit down counter showing the bit time remaining in the current Frame.

Offset: 0x0038

Size (bits): 32

Access: Read only

Reset: 0x0000_0000

Figure 32-37. Frame Remaining Register (OHCI0_HCFMREM)

Bit	Mnemonic	Description	Comments
0	FRT	Frame Remaining Toggle Set whenever FR reaches 0.	Used by the Host Controller driver for synchronization between OHCI0_HCFMINT[FI] and OHCI0_HCFMREM[FR].
1:17		Reserved	
18:31	FR	Frame Remaining Counter is decremented at each bit time. When it reaches 0, it is reset by loading the value specified in OHCI0_HCFMINT[FI] at the next bit time boundary.	When entering the USBOPERATIONAL state, The Host Controller driver loads FR with OHCI0_HCFMINT[FI] and uses the updated value from the next SOF.

32.6.4.33 Frame Number Register (OHCI0_HCFMNUM)

This register is a 16-bit counter. It provides a timing reference among events happening in the Host Controller and the Host Controller driver. The Host Controller driver can use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

Offset: 0x003C

Size (bits): 32

Access: Read only

Reset: 0x0000_0000

Figure 32-38. Frame Number Register (OHCI0_HCFMNUM)

Bit	Mnemonic	Description	Comments
0:15		Reserved	
16:31	FN	Frame Number Incremented when OHCI0_HCFMREM[FR] is loaded.	When entering the USBOPERATIONAL state, this field is incremented automatically. The content is written to HCCA after the Host Controller has incremented FN at each frame boundary and sent a SOF but before the Host Controller reads the first ED in that Frame. After writing to HCCA, the Host Controller sets OHCI0_HCINTSTS[SF].

32.6.4.34 Periodic Start Register (OHCI0_HCPRDSTRT)

This register has a 14-bit programmable value which determines the earliest time the Host Controller should start processing the periodic list.

Offset: 0x0040

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-39. eriodic Start Register (OHCI0_HCPRDSTRT)

Bit	Mnemonic	Description	Comments
0:17		Reserved	
18:31	PS	Periodic Start Cleared by a hardware reset. Set by Host Controller initialization	The value is calculated from OHCI0_HCFMINT[FI]. It can be in error by 10%. A typical value is 0x3E67. When OHCI0_HCFMREM[FR] reaches the value specified, processing of the periodic lists has priority over Control/Bulk processing. The Host Controller starts processing the Interrupt list after completing the current Control or Bulk transaction that is in progress.

32.6.4.35 Low Speed Threahold Register (OHCI0_HCLSTHRES)

This register contains an 11-bit value used by the Host Controller to determine whether to commit to the transfer of a maximum 8-byte LS packet before EOF. Neither the Host Controller nor the Host Controller driver are allowed to change this value.

Offset: 0x0044

Size (bits): 32

Access: R/W

Reset: 0x0000_0628

Figure 32-40. Low Speed Threahold Register (OHCI0_HCLSTHRES)

Bit	Mnemonic	Description	Comments
0:19		Reserved	
20:31	LST	Low Speed Threshold Contains a value which is compared to OHCI0_HCFMREM[FR] prior to initiating a Low Speed transaction. The transaction is started only if OHCI0_HCFMREM[FR] ≥ LST.	This value is calculated by the Host Controller driver and includes consideration of transmission and setup overhead.

32.6.4.36 Root Hub Descriptor A Register (OHCI0_HCRHDESCA)

This register is the first of two (A and B) describing the characteristics of the Root Hub. Reset values are implementation-specific. The descriptor length, descriptor type, and hub controller current fields of the hub Class Descriptor are emulated by the the Host Controller driver. All other fields are located in the OHCI0_HCRHDESCA and OHCI0_HCRHDESCB registers.

Offset: 0x0048

Size (bits): 32

Access: See field specification

Reset: 0x0200_1201

Figure 32-41. Root Hub Descriptor A Register (OHCI0_HCRHDESCA)

Bit	Mnemonic	Description	Comments
0:7	POTPGT	Power-on to Power-Good Time Specifies the duration the Host Controller driver has to wait before accessing a powered-on port of the Root Hub.	R/W The unit of time is 2ms so the duration is calculated as POTPGT x 2ms.
8:18		Reserved	R/W
19	NOCP	No Overcurrent Protection Describes how overcurrent status for the Root Hub ports is reported. 0 Overcurrent status is collectively reported for all downstream ports 1 No overcurrent protection supported	R/W When NOCP = 0, OCPM specifies global or per-port reporting.
20	OCPM	Overcurrent Protection Mode Describes how the overcurrent status for the Root Hub ports are reported. 0 Overcurrent status is reported collectively for all downstream ports 1 Overcurrent status is reported on a per-port basis	R/W OCPM is valid only if NOCP = 0. After reset, OCPM should reflect the same mode as PSM.
21	DT	Device Type Specifies that the Root Hub is not a compound device.	Read on The Root Hub is not permitted to be a compound device so DT should always = 0
22	NPS	No Power Switching Specifies whether power switching is supported or ports are always powered. 0 Ports are power switched 1 Ports are always powered on when the Host Controller is powered on	R/W When NPS = 0, PSM specifies global or per-port switching.

Figure 32-41. Root Hub Descriptor A Register (OHCI0_HCRHDESCA) (continued)

Bit	Mnemonic	Description	Comments
23	PSM	Power Switching Mode Specifies the power switching of the Root Hub ports is controlled. 0 All ports are powered at the same time 1 Each port is powered individually. This mode allows port power to be controlled by either the global switch or the per-port switching. If OHCI0_HCRHDESCB[PPCM] is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, then the port is controlled only by the global power switch (Set/ClearGlobalPower).	R/W This field is valid only if NPS = 0.
24:31	NDP	Number of Downstream Ports These bits specify the number of downstream ports supported by the Root Hub. Minimum = 1 Maximum = 15.	Read only

32.6.4.37 Root Hub Descriptor B Register (OHCI0_HCRHDESCB)

This register is the second of two (A and B) describing the characteristics of the Root Hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

Offset: 0x004C

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-42. Root Hub Descriptor B Register (OHCI0_HCRHDESCB)

Bit	Mnemonic	Description	Comments
0:15	PPCM	Port Power Control Mask Each bit indicates if a port is affected by a global power control command when PSM is set. When 0x0, port is controlled by the global power switch (Set/ClearGlobalPower). Otherwise port's power state is affected only by per-port power control (Set/ClearPortPower). Bit 0: Reserved Bit 1: Ganged-power mask on Port #1 Bit 2: Ganged-power mask on Port #2 ... Bit 15: Ganged-power mask on Port #15	If the device is configured for global switching mode (PSM = 0), this field is not valid.
16:31	DR	Device Removable Each bit is dedicated to a port of the Root Hub. When 0x0, the attached device is removable. Otherwise, the attached device is not removable. Bit 0: Reserved Bit 1: Device attached to Port #1 Bit 2: Device attached to Port #2 ... Bit 15: Device attached to Port #15	

32.6.4.38 Root Hub Status Register (OHCI0_HCRHSTS)

This register is divided into two parts. The lower word represents the Hub Status field and the upper word represents the Hub Status Change field.

Offset: 0x0050

Size (bits): 32

Access: See field specification

Reset: 0x0000_0000

Figure 32-43. Root Hub Status Register (OHCI0_HCRHSTS)

Bit	Mnemonic	Description	Comments
0	CRWE	Clear Remote Wakeup Enable Writing 1 clears DRWE. Writing 0 has no effect.	Read with Clear
1:13		Reserved	
14	OCIC	Overcurrent Indicator Change Set by hardware when a change has occurred to OCI. The Host Controller driver clears this bit by writing 1. Writing 0 has no effect.	R/W
15	LPSC	Local Power Status Change Set to turn power on to all ports (clear OHCI0_HCPRSTS[PPS]) when in global power mode (PSM = 0). In per-port power mode, it sets OHCI0_HCPRSTS[PPS] only on ports whose OHCI0_HCRHDESCB[PPCM] bit is not set. Writing 0 has no effect.	R/W The Root Hub does not support the local power status feature and this bit is always read as 0.
16	DRWE	Device Remote Wakeup Enable Enables a OHCI0_HCPRSTS[CSC] as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 OHCI0_HCPRSTS[CSC] is not a remote wakeup event 1 OHCI0_HCPRSTS[CSC] is a remote wakeup event Writing 1 sets DRWE. Writing 0 has no effect.	R/W
17:29		Reserved	
30	OCI	Overcurrent Indicator This bit is set by hardware when a change has occurred to the OCI field of this register. The Host Controller driver clears this bit by writing 1. Writing 0 has no effect.	R/W
31	LPS	Local Power Status Set to turn off power to all ports (clear OHCI0_HCPRSTS[PPS]) when in global power mode (PSM = 0). In per-port power mode, it clears OHCI0_HCPRSTS[PPS] only on ports whose OHCI0_HCRHDESCB[PPCM] bit is not set. Writing 0 has no effect.	R/W The Root Hub does not support the local power status feature and this bit is always read as 0.

32.6.4.39 Root Hub Port Status Register (OHCI0_HCPRSTS)

This register is used to control and report port events on a per-port basis. *OHCI0_HCRHDESCA*[NDP] represents the number of these registers that are implemented in hardware. The lower word is used to reflect the port status; the upper word reflects the status change bits. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes.

Offset: 0x0054

Size (bits): 32

Access: R/W

Reset: 0x0000_0100

Figure 32-44. Root Hub Port Status Register (OHCI0_HCPRSTS)

Bit	Mnemonic	Description	Comments
0:10		Reserved	
11	PRSC	Port Reset Status Change Indicates end of the 10ms port reset signal. 0 Port reset not complete 1 Port reset complete	The Host Controller driver writes 1 to clear this bit. Writing 0 has no effect.
12	OCIC	Port Overcurrent Indicator Change Root Hub changed POCI. 0 No change in POCI 1 POCI has changed	This bit is valid only if overcurrent conditions are reported on a per-port basis. The Host Controller driver writes 1 to clear this bit. Writing 0 has no effect.
13	PSSC	Port Suspend Status Change Indicates a full resume sequence has been completed. 0 Resume not completed 1 Resume completed	This sequence includes the 20s resume pulse, and the 3ms resychronization delay. The Host Controller driver writes 1 to clear this bit. Writing 0 has no effect. This bit is also cleared when PRSC is set.
14	PESC	Port Enable Status Change Indicates a hardware event cause PES to be cleared. 0 No change in PES 1 Change in PES	Changes from the Host Controller driver do not set this bit. The Host Controller driver writes a 1 to clear this bit. Writing a 0 has no effect.
15	CSC	Connect Status Change Indicates a connect or disconnect event occured. 0 No change in CCS 1 Change in CCS	The Host Controller driver writes 1 to clear this bit. Writing 0 has no effect. If CCS is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected. **Note:** If OHCI0_HCRHDESCB[DR] is set, this bit is set only after a Root Hub reset to inform the system that the device is attached.
16:21		Reserved	
22	LSDA	Low Speed Device Attached Indicates the speed of the device attached to this port 0 Full-speed device attached 1 Low-speed device attached	This field is valid only when CCS is set. The Host Controller driver clears the PPS by writing 1 to this bit. Writing 0 has no effect.

Figure 32-44. Root Hub Port Status Register (OHCI0_HCPRSTS) (continued)

Bit	Mnemonic	Description	Comments
23	PPS	Port Power Status Reflects the port's power status 0 Port power is off 1 Port power is on	This bit is cleared if an overcurrent condition is detected. Which power control switches are enabled is determined by OHCI0_HCRHDESCA[PSM] and OHCI0_HCRHDESCB[PPCM]. In global switching mode (PSM = 0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PSM = 1), if OHCI0_HCRHDESCB[PPCM] for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CCS, PES, PSS, and PRS should be reset. The Host Controller driver writes 1 to set the this bit. Writing 0 has no effect. **Note:** This bit is always reads 1 if power switching is not supported.
24:26		Reserved	
27	PRS	Port Reset Status Indicates status of port reset signal. 0 Port reset signal is not active 1 Port reset signal is active	When this bit is set, port reset signaling is asserted. When reset is completed, this bit is cleared when PRSC is set. This bit cannot be set if CCS is cleared. The Host Controller driver sets the port reset signaling by writing 1 to this bit. Writing 0 has no effect. If CCS is cleared, this write does not set this bit, but instead sets CSC. This informs the driver that it attempted to reset a disconnected port.
28	POCI	Port Overcurrent Indicator Reflects the overcurrent input signal 0 No overcurrent condition 1 Overcurrent condition detected	This bit is only valid when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. The Host Controller driver writes 1 to initiate a resume. Writing 0 has no effect. A resume is initiated only if PSS is set.
29	PSS	Port Suspend Status Indicates the port is suspended or in the resume sequence. 0 Port is not suspended 1 Port is suspended	PSS is set by a port suspend and cleared when PSSC is set at the end of the resume interval. This bit cannot be set if CCS is cleared. This bit is also cleared when PRSC is set at the end of the port reset or when the Host Controller is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the Host Controller. The Host Controller driver sets PSS bit by writing 1. Writing 0 has no effect. If CCS is cleared, this write does not set PPSS; instead it sets CSC. This informs the driver that it attempted to suspend a disconnected port.

Figure 32-44. Root Hub Port Status Register (OHCI0_HCPRSTS) (continued)

Bit	Mnemonic	Description	Comments
30	PES	Port Enable Status Indicates whether the port is enabled or disabled. 0 Port is disabled 1 Port is enabled	The Root Hub can clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error such as babble is detected. This change also causes PESC to be set. PES cannot be set when CCS is cleared. PES is also set at the completion of a port reset when PRSC is set, or a port suspend when PSSC is set. The Host Controller driver sets PES by writing 1. Writing 0 has no effect. If CCS is cleared, this write does not set PES, but instead sets CCS. This informs the driver that it attempted to enable a disconnected port.
31	CCS	Current Connect Status Reflects the current state of the downstream port. 0 No device connected 1 Device connected The Host Controller driver writes 1 to this bit to clear PES. Writing a '0' has no effect.	This bit is always read as 1 when the attached device is non-removable

32.6.4.40 Bridge Control DCR (USB2HMDCR0_BCNTL)

This register controls the configuration of the AHB Slave-to-OPB Master Bridge for the USB Host Controller function.

Offset: 0x0360

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-45. Bridge Control DCR (USB2HMDCR0_BCNTL)

Bit	Mnemonic	Description	Comments
0	DCRUM	DCR User Mode Enable. 0 Privilege Mode (default). 1 User Mode.	When this bit is set, the DCR slave of the AHB-to-OPB Bridge allows nonpriviledge access to the bridge DCR registers. This bit can only be set in Privilege Mode.
1	DISLCK	Disable OPB Bus Lock Operation. 0 Lock transfers on the OPB are allowed (default). 1 Disable lock transfers on the OPB bus.	
2:31		Reserved	

32.6.4.41 Bridge Status DCR (USB2HMDCR0_BSTAT)

This register contains status and errors reported during operation of the OPB Master contained within the AHB Slave-to-OPB Master Bridge for the USB Host Controller function.

Offset: 0x0361

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-46. Bridge Status DCR (USB2HMDCR0_BSTAT)

Bit	Mnemonic	Description	Comments
0	OTOE	OPB Timeout Error	This bit is set when the bridge encounters a timeout on the OPB bus.
1	OEACK	OPB Error Acknowledge	This bit is set when the bridge encounters an Error Acknowledge on the OPB bus.
2:31		Reserved	

Programming Note: Do not attempt to read or write registers at offset 2, 3, 4, or 5.

32.6.4.42 Bridge OPB Latency DCR (USB2HMDCR0_LAT)

This register contains the enable and count value for the OPB master latency counter contained of the AHB Slave-to-OPB Master Bridge for the USB Host Controller function.

Offset: 0x0366

Size (bits): 32

Access: R/W

Reset: 0x0000_0000

Figure 32-47. Bridge OPB Latency DCR (USB2HMDCR0_LAT)

Bit	Mnemonic	Description	Comments
0:3	LATCNT	OPB Latency Count Value. 0000 Disable Latency Counter. Master may own bus indefinitely. (default) 0001 1 x 16 = 16 0010 2 x 16 = 32 0011 3 x 16 = 48 0100 4 x 16 = 64 0101 5 x 16 = 80 0110 6 x 16 = 96 0111 7 x 16 = 112 1000 8 x 16 = 128 1001 9 x 16 = 144 1010 10 x 16 = 160 1011 11 x 16 = 176 1100 12 x 16 = 192 1101 13 x 16 = 208 1110 14 x 16 = 224 1111 15 x 16 = 240	Number of XferAcks is longest locked transfer if OPB_pendReqn = 1
4:31		Reserved	

32.6.4.43 Bridge Revision ID DCR (USB2HMDCR0_REVID)

Reading this register returns the revision ID of the AHB Slave-to-OPB Master Bridge for the USB20HOSTOPBBR1 fumction. The values for the Revision Number and the Branch ID are subject to change.

Note: Attempting to write this register does not cause an error, as the data in it remains unchanged.

Offset: 0x0367

Size (bits): 32

Access: Read only

Reset: 0x0000_0110

Figure 32-48. Bridge Revision ID DCR (USB2HMDCR0_REVID)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12:23	RID	Revision number	
24:31	BRID	Branch Revision ID	

32.6.4.44 Bridge Control DCR (USB2HSDCR0_BCNTL)

This register controls the configuration of the OPB Slave-to-AHB Master bridge for the USB20HOSTOPBBR1 function.

Offset: 0x0370

Size (bits): 32

Access: R/W

Reset: 0x0D00_0110

Figure 32-49. Bridge Control DCR (USB2HSDCR0_BCNTL)

Bit	Mnemonic	Description	Comments
0	DCRUM	DCR User Mode Enable. 0 Priviledge Mode (default). 1 User Mode.	When this bit is set, the DCR slave of the OPB-to-AHB bridge allows nonpriviledge access to the bridge DCR registers. This bit can only be set in Privilege Mode.
1:3		Reserved	
4:7	RTYCNL	OPB Retry Control Count Number of cycles to wait for HGRANT before an OPB Retry is issued. 0000 1 0001 2 0010 3 0011 4 0100 5 0101 6 0110 7 0111 8 1000 9 1001 10 1010 11 1011 12 1100 13 1101 14 (Default) 1110 15 1111 16	To alleviate deadlock scenarios on the OPB bus the OPB-to-AHB bridge issues an OPB Retry; this retry forces the requesting OPB master to abandon the pending data transfer, and allow the OPB arbiter to rearbitrate the bus. These bits allows the user to program the bridge OPB Retry response from an AHB HGRANT reference; they indicate the number of cycles to wait for HGRANT before an OPB Retry is issued, and the AHB request is terminated.
8:31		Reserved	

Programming Note: Do not attempt to read or write registers at offset 1 or 2.

32.6.4.45 Bridge Revision ID DCR (USB2HSDCR0_REVID)

Reading this register returns the revision ID of the OPB Slave-to-AHB Master bridge for the USB20HOSTOPBBR1 function. The values for the Revision Number and the Branch ID are subject to change.

Note: Attempting to write this register does not cause an error, as the data in it remains unchanged.

Offset: 0x0373

Size (bits): 32

Access: Read only

Reset: 0x0000_0100

Figure 32-50. Bridge Revision ID DCR (USB2HSDCR0_REVID)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12:23	RID	Revision number	
24:31	BRID	Branch Revision ID	

32.7 USB 2.0 Device Interface

The PPC440EPx Embedded Processor implements the Universal Serial Bus (USB) version 2.0 high speed (480 Mbps) and full speed (12 Mbps) device functions. The USB20DEVICE core has support for three IN Endpoints and three OUT Endpoints in addition to the required Endpoint 0. Endpoint 0 supports the required Control type transfers, while the additional Endpoints can be software controlled to support Bulk, Interrupt, Isochronous transfer types, or interrupt requests for FIFO service. Endpoints 1 and 2 can also be software controlled to select DMA.

System access for the USB 2.0 Device support is provided through the OPB 32-bit slave interface. The USB cable connection is provided through an 8-bit Universal Transceiver Macrocell Interface (UTMI) for high- and full-speed implementations. The UTMI interface is used to connect a UTMI+ compliant off-chip or on-chip PHY. This core incorporates the configurable, synthesizable, Mentor Graphics Inventra™ library USB 2.0 High/Full Speed Function controller and a single one-port SRAM array and its associated BIST controller for implementing all data storage FIFOs. *Figure 32-52* on page 865 shows the system implementation.

32.7.1 Features

- USB 2.0 high- and full-speed device controller.
- 32-bit OPB slave interface.
- Provides an 8-bit 60 MHz UTM+I with an interface for high- and full-speed operation.
- DMA request support for Endpoints 1 and 2.
- Supports a total of four Endpoints in each direction, including the required control Endpoint 0.
- Endpoints 1–3 can be IN only, OUT only, or IN and OUT.
- Endpoints 1–3 can be configured to support either Interrupt/Bulk-only transfer types, software programmable Interrupt/Bulk, or Isochronous transfer types.
- High bandwidth isochronous transfers are supported.

Figure 32-51. Typical USB 2.0 Device Interface System Application Block Diagram

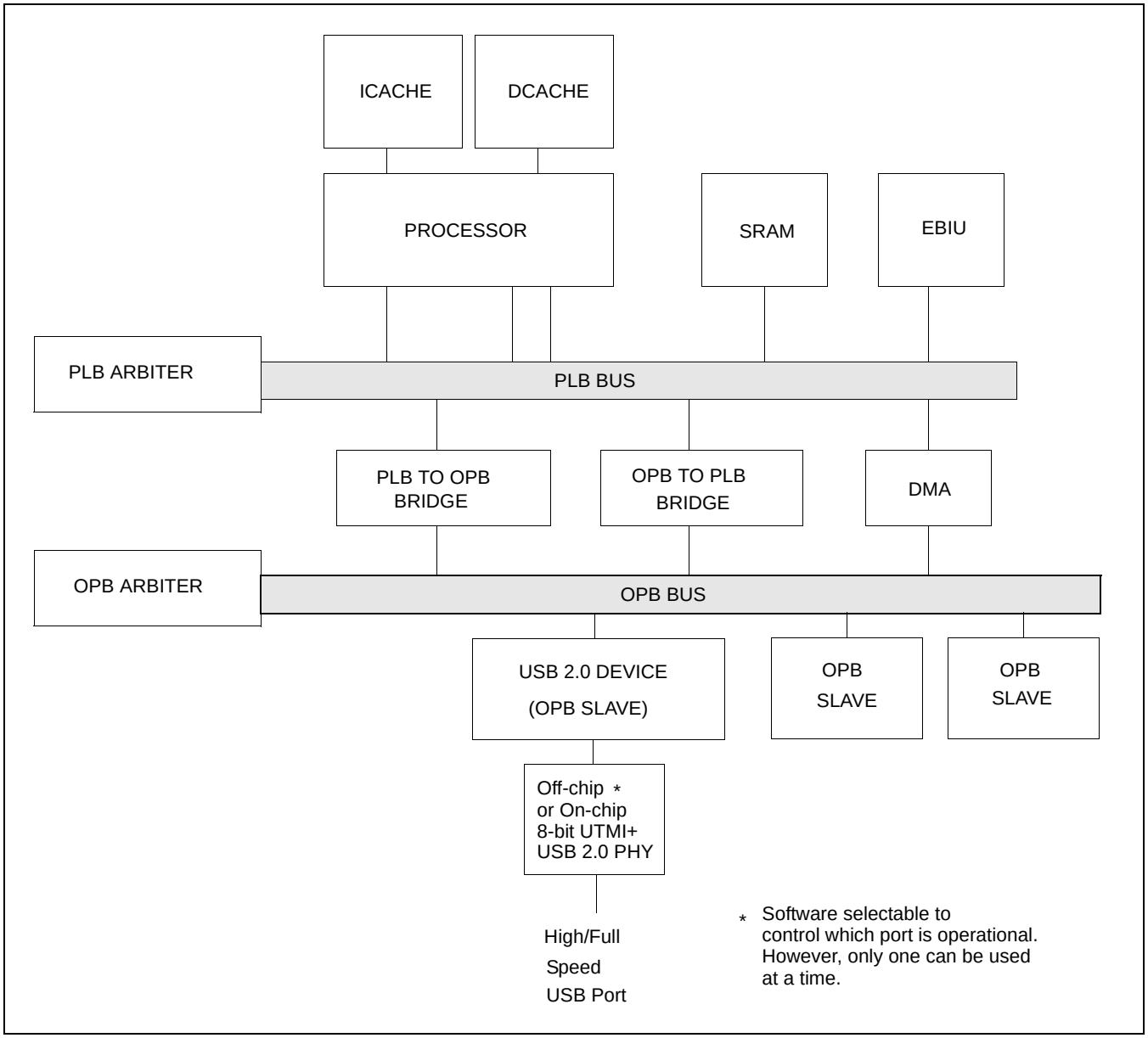
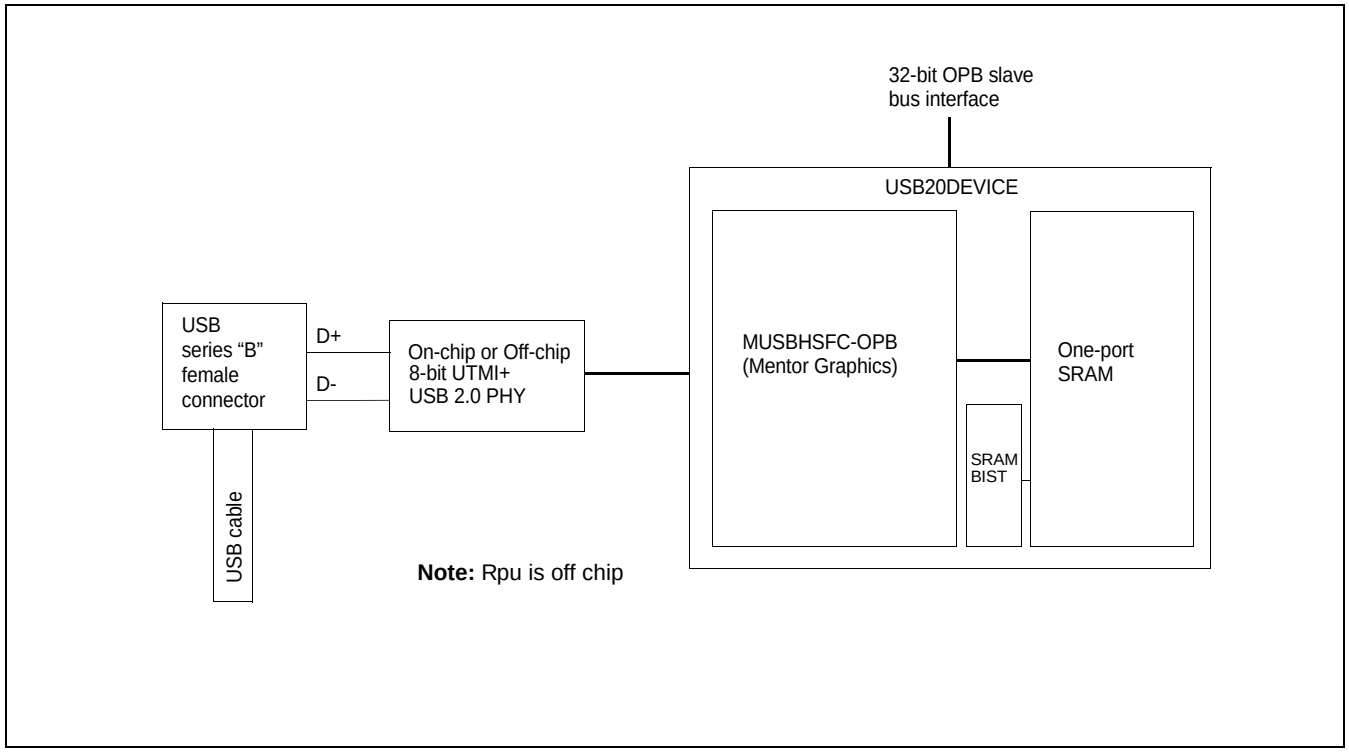


Figure 32-52. USB 2.0 Device Interface System Block Diagram



32.7.2 Software Interface

All internal registers for the USB20DEVICE core are contained within the Mentor Graphics Inventra Library USB 2.0 High/Full Speed Function Controller core. Because of this, the Mentor Graphic core's product specification and programmer's guide can be referenced for register and programming information. Refer to *Section 32.1.1 References* on page 809 for these documents.

The following sections provide the required detailed information concerning the USB 2.0 Device support register set as it is instantiated in the PPC440EPx chip.

32.7.3 Memory Map Definitions

The USB 2.0 Device support memory map is divided into three sections:

1. USB Device Registers.
   These registers provide control and status for the entire function.
2. Indexed Registers.
   These registers provide control and status for one Endpoint. There is an USB2D0_INMAXP and USB2D0_INCSR register for each IN Endpoint in the USB20DEVICE and a USB2D0_OUTMAXP, USB2D0_OUTCSR, and USB2D0_OUTCOUNT for each OUT Endpoint (except for Endpoint 0 which has a reduced registered set). Only the registers for one IN Endpoint and one OUT Endpoint appear in the register map at any one time. The Endpoints are selected by writing the Endpoint number to the USB2D0_INDEX register. Therefore, to access the registers for IN Endpoint 1 and OUT Endpoint 1, 1 must first be written to the USB2D0_INDEX register and then the control and status registers appear in the memory map. See *Indexed Registers* on page 876.

3. FIFOs.
   The FIFOs for each IN Endpoint appear as a 32-bit double words consecutively in the memory map. The FIFOs for each OUT Endpoint also appear consecutively at the same set of addresses. See *FIFOs* on page 885.

The following sections identify and classify the registers and FIFOs, and provide size and access information.

32.7.4 USB 2.0 Device Registers

Table 32-5 lists the registers provided for the USB 2.0 Device function. The following sections define the bit-level detail for the registers.

.

Table 32-5. USB Device Registers

Register Name	Description	Address	Size (bits)	Access	Page
USB2D0_INTRIN	Interrupts for Endpoints 0 and IN 1–3	0x0 E000 0100	16	R	867
USB2D0_POWER	Power management	0x0 E000 0102	8	R/W	868
USB2D0_FADDR	Function address	0x0 E000 0103	8	See register	869
USB2D0_INTRINE	Interrupt enable for USB2D0_INTRIN	0x0 E000 0104	16	R/W	870
USB2D0_INTROUT	Interrupts for OUT Endpoints 1 to 3	0x0 E000 0106	16	R	870
USB2D0_INTRUSBE	Interrupt enable for USB2D0_INTRUSB	0x0 E000 0108	8	R/W	871
USB2D0_INTRUSB	Interrupts for common USB	0x0 E000 0109	8	R	872
USB2D0_INTROUTE	Interrupt enable for USB2D0_INTROUT	0x0 E000 010A	16	R/W	873
USB2D0_TSTMODE	Enable the USB 2.0 test modes	0x0 E000 010C	8	R/W	874
USB2D0_INDEX	Endpoint Status/Control Selection Index	0x0 E000 010D	8	R/W	875
USB2D0_FRAME	Frame Number	0x0 E000 010E	16	R	875

32.7.4.1 Interrupts for Endpoints 0 and IN 1–3 Register (USB2D0_INTRIN)

USB2D0_INTRIN indicates which of the interrupts for IN Endpoints 1 – 3 are currently active. It also indicates whether the Endpoint 0 interrupt is currently active.

Note: Bits relating to Endpoints that have not been configured will always return 0. All active interrupts are cleared when this register is read.

Address: 0x0 E0000100

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 32-53. Interrupts for Endpoints 0 and IN 1–3 Register (USB2D0_INTRIN)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	IN Endpoint 3 interrupt	
13	EP2	IN Endpoint 2 interrupt	
14	EP1	IN Endpoint 1 interrupt	
15	EP0	Endpoint 0 interrupt	

32.7.4.2 Power Management Register (USB2D0_POWER)

USB2D0_POWER controls Suspend and Resume signaling, and high-speed operation.

Address: 0x0 E0000102

Size (bits): 8

Access: Read/Write

Reset: 0x20

Figure 32-54. Power Management Register (USB2D0_POWER)

Bit	Mnemonic	Description	Comments
0	ISOU	ISO Update 0 Do not wait for SOF token. 1 Wait for an SOF token from the time IPR is set before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.	The ISOU bit only affects Endpoints performing Isochronous transfers. It affects all IN Isochronous Endpoints and is normally used as a method of ensuring a "clean" start-up of an IN Isochronous pipe.
1		Reserved	
2	HSE	High-Speed Enable 0 Device only operates in Full-Speed mode. 1 Negotiate for High-Speed mode when the device is reset by the hub.	The HSE bit is used to disable High-Speed operation. Normally the interface will automatically negotiate for high-speed operation, when it is reset, by sending a "chirp" to the hub. However if this bit is cleared then the interface will not send any "chirps" to the hub so the function will remain in Full-Speed mode, even when connected to a High-Speed capable USB.
3	HSM	High-Speed Mode 0 Interface is not operating in High-Speed mode. 1 Interface has successfully negotiated for High-Speed mode.	The HSM bit is used to determine whether the interface is in High- or Full-Speed mode. It set to 1 when the function has successfully negotiated for high-speed operation during a USB reset.
4	RST	Reset 0 Reset signaling is not present on the bus. 1 Reset signaling is present on the bus.	The RST bit is used to determine when reset signaling is present on the USB. It is set to 1 when reset signaling is detected and remains high until the bus reverts to an idle state.
5	RSM	Resume 0 Do not generate Resume signaling. 1 Generate Resume signaling when the function is in Suspend mode.	The RSM bit is used to generate Resume signaling on the USB to perform remote wake-up from Suspend mode. Once set, it should be left high for approximately 10 ms (at least 1ms and no more than 15ms), then cleared.
6	SUSMDE	Suspend Mode 0 Exiting Suspend mode 1 Entering Suspend mode	The Suspend Mode bit is set to 1 when Suspend mode is entered. It is cleared when the USB2D0_INTRUSB register is read (as a result of receiving a Suspend interrupt). It will also be cleared if Suspend mode is exited by setting the RSM bit to initiate a remote wake-up.
7	ESUSPM	Enable Suspend Mode 0 Disable the SUSPENDM signal. 1 Enable the SUSPENDM signal.	The ESUSPM bit is set to enable the SUSPENDM signal to put the UTM (and any other hardware which uses the SUSPENDM signal) into Suspend mode. If this bit is not set, Suspend mode is detected as normal but the SUSPENDM signal remains high so that the UTM does not go into its low-power mode.

32.7.4.3 Function Address Register (USB2D0_FADDR)

USB2D0_FADDR is written with the function's 7-bit address (received through a SET_ADDRESS descriptor). It is then used for decoding the function address in subsequent token packets.

This register should be written with the address value contained in the SET_ADDRESS standard device request (see USB Specification Revision 2.0, Chapter 9), when it is received on Endpoint 0. The new address will not take effect immediately as the processor will still be using the old address for the Status stage of the device request. The function will continue to use the old address for decoding packets until the device request has completed. The status of the device request can be determined by reading bit 0 of this register. When a new address is written to this register, bit 0 is automatically set. It remains high until the device request has completed and is cleared when the new address takes effect.

Note: While the firmware may write the new address to the FADDR register immediately, it is recommended that this operation be performed in the Status phase of the operation in case the processor aborts the command.

Address: 0x0 E0000103

Size (bits): 8

Access: R/W except bit 0 which is read-only

Reset: 0x00

Figure 32-55. Function Address Register (USB2D0_FADDR)

Bit	Mnemonic	Description	Comments
0	UPD	Update 0 Cleared when the new address takes effect (at the end of the current transfer). 1 Set when FADDR is written.	
1:7	FADDR	Function Address	

32.7.4.4 Interrupt Enable for USB2D0_INTRIN Register (USB2D0_INTRINE)

USB2D0_INTRINE provides interrupt enable bits for each of the interrupts in USB2D0_INTRIN. On reset, EP0 – EP3 In are set to 1 where n is the number of IN Endpoints (in addition to Endpoint 0) that are included in the design, while the remaining bits are set to 0.

Note: Bits relating to Endpoints that have not been configured will always return 0.

Address: 0x0 E0000104

Size (bits): 16

Access: R/W

Reset: 0xFF masked with the IN Endpoints implemented

Figure 32-56. Interrupt Enable for USB2D0_INTRIN Register (USB2D0_INTRINE)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	Enable IN Endpoint 3 interrupt	
13	EP2	Enable IN Endpoint 2 interrupt	
14	EP1	Enable IN Endpoint 1 interrupt	
15	EP0	Enable Endpoint 0 interrupt	

32.7.4.5 Interrupts for OUT Endpoints 1 to 3 Register (USB2D0_INTROUT)

USB2D0_INTROUT indicates which of the interrupts for OUT Endpoints 1 – 3 are currently active. (Endpoint 0 uses a single interrupt, included in the USB2D0_INTRIN register.) Note: Bits relating to Endpoints that have not been configured will always return 0.

Note: Bits relating to Endpoints that have not been configured will always return 0. All active interrupts are cleared when this register is read.

Address: 0x0 E0000106

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 32-57. Interrupts for OUT Endpoints 1 to 3 Register (USB2D0_INTROUT)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	OUT Endpoint 3 interrupt	
13	EP2	OUT Endpoint 2 interrupt	
14	EP1	OUT Endpoint 1 interrupt	
15		Reserved	

32.7.4.6 Interrupt enable for USB2D0_INTRUSB Register (USB2D0_INTRUSBE)

USB2D0_INTRUSBE provides interrupt enable bits for each of the interrupts in USB2D0_INTRUSB.

Address: 0x0 E0000108

Size (bits): 8

Access: R/W

Reset: 0x06

Figure 32-58. Interrupt enable for USB2D0_INTRUSB Register (USB2D0_INTRUSBE)

Bit	Mnemonic	Description	Comments
0:3		Reserved	
4	SOF	Start of Frame 0 Disable SOF interrupt in USB2D0_INTRUSB. 1 Enable SOF nterrupt in USB2D0_INTRUSB.	
5	RST	Reset 0 Reset signaling not detected on the bus. 1 Reset signaling detected on the bus.	
6	RSM	Resume 0 Resume signaling not detected on the bus while in Suspend mode. 1 Resume signaling detected on the bus while in Suspend mode.	
7	SUSP	Suspend 0 Suspend signaling not detected on the bus. 1 Suspend signaling detected on the bus.	

32.7.4.7 Interrupts for Common USB Register (USB2D0_INTRUSB)

USB2D0_INTRUSB indicates which USB interrupts are currently active.

Note: All active interrupts are cleared when this register is read.

Address: 0x0 E0000109

Size (bits): 8

Access: R/W

Reset: 0x00

Figure 32-59. Interrupts for Common USB Register (USB2D0_INTRUSB)

Bit	Mnemonic	Description	Comments
0:3		Reserved	
4	SOF	Start of Frame Set at the SOF.	
5	RST	Reset 0 Reset signaling not detected on the bus. 1 Reset signaling detected on the bus.	
6	RSM	Resume 0 Resume signaling not detected on the bus while in Suspend mode. 1 Resume signaling detected on the bus while in Suspend mode.	
7	SUSP	Suspend 0 Suspend signaling not detected on the bus. 1 Suspend signaling detected on the bus.	

32.7.4.8 Interrupt Enable for USB2D0_INTROUT Register (USB2D0_INTROUTE)

USB2D0_INTROUTE provides interrupt enable bits for each of the interrupts in USB2D0_INTROUT. On reset, EP1 Out – EP*m* Out are set to 1 where *m* is the number of OUT Endpoints (in addition to Endpoint 0) that are included in the design, while the remaining bits are set to 0.

Note: Bits relating to Endpoints that have not been configured will always return 0.

Address: 0x0 E000010A

Size (bits): 16

Access: R/W

Reset: 0xFF masked with the OUT Endpoints implemented

Figure 32-60. Interrupt Enable for USB2D0_INTROUT Register (USB2D0_INTROUTE)

Bit	Mnemonic	Description	Comments
0:11		Reserved	
12	EP3	Enable OUT Endpoint 3 interrupt	
13	EP2	Enable OUT Endpoint 2 interrupt	
14	EP1	Enable OUT Endpoint 1 interrupt	
15		Reserved	

32.7.4.9 Enable the USB 2.0 Test Modes Register (USB2D0_TSTMODE)

USB2D0_TSTMODE puts the function into one of the four test modes for High-Speed operation described in the USB 2.0 specification. It is not used in normal operation.

Note: Only one of bits 2:7 should be set at any time.

Address: 0x0 E000010C

Size (bits): 8

Access: R/W

Reset: 0x00

Figure 32-61. Enable the USB 2.0 Test Modes Register (USB2D0_TSTMODE)

Bit	Mnemonic	Description	Comments
0:1		Reserved	
2	FFS	Force Full-Speed Force the interface into Full-Speed mode	Set this bit to force the interface into Full-Speed mode when a USB reset is received.
3	FHS	Force High-Speed Force the interface into High-Speed mode	Set this bit to force the interface into High-Speed mode when a USB reset is received.
4	TESTP	Test Packet Enter Test Packet test mode.	In HS mode, set this bit to enter the Test Packet test mode. In this mode, the interface repetitively transmits a 53-byte test packet, whose form is defined in the USB Specification Revision 2.0, Section 7.1.20. **Note:** The test packet has a fixed format must be loaded into the Endpoint 0 FIFO before the test mode is entered.
5	TESTK	Test K Enter the Test K test mode	In HS mode, set this bit to enter the Test K test mode. In this mode, the interface transmits a continuous K on the bus.
6	TESTJ	Test J Enter the Test J test mode	In HS mode, set this bit to enter the Test J test mode. In this mode, the interface transmits a continuous J on the bus.
7	TSTSE0	Test SE0 NAK Enter the Test_SE0_NAK test mode	In HS mode, set this bit to enter the Test SE0 NAK test mode. In this mode, the interface remains in High-Speed mode but responds to any valid IN token with a NAK.

32.7.4.10 Endpoint Status/Control Selection Index Register (USB2D0_INDEX)

USB2D0_INDEX determines which Endpoint control/status registers (indexed registers) are accessed at addresses 0x0 E0000110 to 0x0 E0000119.

Each IN Endpoint and each OUT Endpoint have their own set of control/status registers. Only one set of IN control/status and one set of OUT control/status registers appear in the memory map at any one time. Before accessing an Endpoint's control/status registers, the Endpoint number should be written to the USB2D0_INDEX register to ensure that the correct control/status registers appear in the memory map.

Address: 0x0 E000010D

Size (bits): 8

Access: R/W

Reset: 0x00

Figure 32-62. Endpoint Status/Control Selection Index Register (USB2D0_INDEX)

Bit	Mnemonic	Description	Comments
0:3		Reserved	
4:7		Selected Endpoint	Bit 4 is the msb.

32.7.4.11 Frame Number Register (USB2D0_FRAME)

USB2D0_FRAME holds the last received frame number.

Address: 0x0 E000010E

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 32-63. Frame Number Register (USB2D0_FRAME)

Bit	Mnemonic	Description	Comments
0:3		Always 0.	
4:15		Frame Number	Bit 4 is the msb.

32.7.5 Indexed Registers

There are six Endpoints in addition to Endpoint 0, three IN and three OUT. Their registers are accessed indirectly using the USB2D0_INDEX register. The number of the Endpoint must first be written to USB2D0_INDEX, then subsequent reads or writes to function specific registers (USB2D0_INCSR, USB2D0_INMAXP, USB2D0_OUTCSR, USB2D0_OUTMAXP, USB2D0_OUTCOUNT) return (or update) the information only for the Endpoint specified by the USB2D0_INDEX register.

Each of the three IN Endpoints implements two registers, USB2D0_INCSR and USB2D0_INMAXP. Note that the CSR function is defined differently depending on the Endpoint. Endpoint 0 implements only eight bits (and is called USB2D0_INCSR0), whereas Endpoints 1 to 3 implement 16 bits (referred to as USB2D0_INCSR). This explains why address 0x0 E0000110 appears twice in the table.

Similarly, each of the three OUT Endpoints implement three registers, USB2D0_OUTCSR, USB2D0_OUTMAXP and USB2D0_OUTCOUNT. Note that the COUNT function is defined differently depending on the Endpoint. Endpoint 0 implements only eight bits (and is called USB2D0_OUTCOUNT0), whereas Endpoints 1 to 3 implement 16 bits (referred to as USB2D0_OUTCOUNT). This explains why address 0x0 E000011A appears twice in the table.

Table 32-6. Indexed Registers

Register Name	Description	Address	Size (bits)	Access	Page
USB2D0_INCSR0	Control/status register for Endpoint 0. (Index register set to select Endpoint 0)	0x0 E000 0110	8	R/W	877
USB2D0_INCSR	Control/status register for IN Endpoint. (Index register set to select Endpoints 1–3)	0x0 E000 0110	16	R/W	878
USB2D0_INMAXP	Maximum packet size for IN Endpoint. (Index register set to select Endpoints 1–3)	0x0 E000 0112	16	R/W	880
USB2D0_OUTCSR	Control/Status register for OUT Endpoint. (Index register set to select Endpoints 1–3)	0x0 E000 0114	16	R/W	881
USB2D0_OUTMAXP	Maximum packet size for OUT Endpoint. (Index register set to select Endpoints 1–3)	0x0 E000 0116	16	R/W	883
USB2D0_OUTCOUNT0	Number of received bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)	See register description	8	R	884
USB2D0_OUTCOUNT	Number of bytes in OUT Endpoint FIFO. (Index register set to select Endpoints 1–3)	0x0 E000 011A	16	R	884

32.7.5.1 Endpoint 0 Control Status Register (USB2D0_INCSR0)

USB2D0_INCSR0 provides control and status bits for Endpoint 0.

Address: 0x0 E0000110 (0x0 E0000111 when the Index register is set to 0)

Size (bits): 8

Access: R/W

Reset: 0x00

Figure 32-64. Endpoint 0 Control Status Register (USB2D0_INCSR0)

Bit	Mnemonic	Description	Comments
0	SSE	Serviced Setup End 0 SE bit is clear 1 Clear the SE bit	The processor writes 1 to this bit to clear the SE bit. SSE is cleared automatically after the operation is complete.
1	SOPR	Serviced Out Packet Ready 0 OPR bit is clear 1 Clear the OPR bit.	The processor writes 1 to this bit to clear the OPR bit. SOPR is cleared automatically after the operation is complete.
2	SS1	Send Stall 0 Terminate the STALL condition 1 Transmit a STALL handshake	The processor writes 1 to this bit to issue a STALL handshake. This bit is automatically cleared.
3	SE	Setup End 0 SSE bit set 1 Control tansaction ended before DE bit set	This bit is set when a control transaction ends before the DE bit has been set. An interrupt is generated and the FIFO flushed at this time. The bit is cleared by the processor writing 1 to the SSE bit.
4	DE	Data End 0 Last data packet processed 1 Last data packet	The processor sets this bit: 1. When setting IPR for the last data packet. 2. When clearing OPR after unloading the last data packet. 3. When setting IPR for a zero length data packet. It is cleared automatically.
5	SS0	Sent Stall 0 STALL terminated 1 STALL handshake has been transmitted	This bit is set when a STALL handshake has been transmitted. The processor should clear this bit.
6	IPR	In Packet Ready 0 Data packet in FIFO transmitted 1 Data packet loaded in FIFO	The processor sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when this bit is cleared.
7	OPR	Out Packet Ready 0 SOPR bit set 1 Data packet received	This bit is set when a data packet has been received. An interrupt is generated (if enabled) when this bit is set. The processor clears this bit by setting the SOPR bit.

32.7.5.2 Control/Status for IN Endpoint Register (USB2D0_INCSR)

USB2D0_INCSR provides control and status bits for IN transactions through the currently selected Endpoint. There is an USB2D0_INCSR register for each IN Endpoint (not including Endpoint 0).

Address: 0x0 E0000110

Size (bits): 16

Access: R/W

Reset: 0x00

Figure 32-65. Control/Status for IN Endpoint Register (USB2D0_INCSR)

Bit	Mnemonic	Description	Comments
0	AUTOS	Automatically Set 0 Do not set IPR 1 Set IPR automatically	If the processor sets this bit, IPR is automatically set when the maximum packet size (value in USB2D0_INMAXP) is loaded into the IN FIFO. If a packet of less than the maximum packet size is loaded, IPR must be set manually. **Note:** This bit should not be set for High-Bandwidth Isochronous Endpoints.
1	ISO	ISO Mode 0 Bulk/Interrupt transfer mode 1 Isochronous transfer mode	The processor sets this bit to enable the IN Endpoint for Isochronous transfers (ISO mode), and clears it to enable the IN Endpoint for Bulk/ Interrupt transfers.
2	MODE	Mode 0 Endpoint direction is OUT 1 Endpoint direction is IN	The processor sets this bit to enable the Endpoint direction as IN, and clears it to enable the Endpoint direction as OUT. **Note:** This bit valid only when the Endpoint FIFO is used for both IN and OUT transactions, otherwise it is ignored.
3	DMAE	DMA Enable 0 Disable the DMA request for the IN Endpoint 1 Enable the DMA request for the IN Endpoint	The processor sets this bit to enable the DMA request for the IN Endpoint.
4	FDT	Force Data Toggle 0 IN data toggle switches after receiving an ACK 1 IN data toggle switches after each data packet is sent	The processor sets this bit to force the Endpoint's IN data toggle to switch after each data packet is sent regardless of whether an ACK was received. This can be used by Interrupt IN Endpoints which are used to communicate rate feedback for Isochronous Endpoints.
5:7		Reserved	
8	ITX	Incomplete Transmission 0 Not High-Bandwidth Isochronous transfer 1 Insufficient IN tokens received for split packet transmission during High-Bandwidth Isochronous transfer	When the Endpoint is being used for High-Bandwidth Isochronous transfers, this bit is set to indicate where a large packet has been split into two or three packets for transmission, but insufficient IN tokens have been received to send all the parts. **Note:** In anything other than a High-Bandwidth Isochronous transfer, this bit will always return 0.
9	CDT	Clear Data Toggle 0 Operation to reset the IN Endpoint data toggle complete 1 Reset the IN Endpoint data toggle	The processor writes 1 to this bit to reset the IN Endpoint data toggle.

Figure 32-65. Control/Status for IN Endpoint Register (USB2D0_INCSR) (continued)

Bit	Mnemonic	Description	Comments
10	SS0	Sent Stall 0 STALL terminated 1 STALL handshake has been transmitted	This bit is set when a STALL handshake has been transmitted. The FIFO is flushed and the IPR bit is cleared. The processor should clear this bit.
11	SS1	Send Stall 0 Terminate the STALL condition 1 Transmit a STALL handshake to an IN token	The processor writes 1 to this bit to issue a STALL handshake to an IN token. The processor clears this bit to terminate the stall condition. **Note:** This bit has no effect where the Endpoint is being used for Isochronous transfers.
12	FFIFO	Flush FIFO 0 Flush complete 1 Flush the next packet to be transmitted from the Endpoint IN FIFO	The processor writes 1 to this bit to flush the next packet to be transmitted from the Endpoint IN FIFO. The FIFO pointer is reset and the IPR bit is cleared. **Note 1.** FFIFO has no effect when written before the IPR bit is set. Set IPR and then set FFIFO with two separate writes to USB2D0_INCSR. **Note 2.** If the FIFO contains two packets, FFIFO will need to be set twice to completely clear the FIFO.
13	UR	Underrun 0 IN underrun processed 1 IN data underrun	In ISO mode, this bit is set when a zero-length data packet is sent after receiving an IN token with the IPR bit not set. In Bulk/Interrupt mode, this bit is set when a NAK is returned in response to an IN token. The processor should clear this bit.
14	FIFONE	FIFO Not Empty 0 No packet in the IN FIFO 1 At least one packet in the IN FIFO	This bit is set when there is at least one packet in the IN FIFO.
15	IPR	In Packet Ready 0 Space available in the FIFO 1 Data packet loaded into the FIFO	The processor sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. If the FIFO is double-buffered, it is also automatically cleared when there is space for a second packet in the FIFO. An interrupt is generated (if enabled) when the bit is cleared.

32.7.5.3 Maximum Packet Size for IN Endpoint Register (USB2D0_INMAXP)

USB2D0_INMAXP defines the maximum amount of data that can be transferred through the selected IN Endpoint in a single frame/microframe (High-Speed transfers). There is an USB2D0_INMAXP register for each IN Endpoint (except Endpoint 0).

Address: 0x0 E0000112

Size (bits): 16

Access: R/W

Reset: 0x00

Figure 32-66. Maximum Packet Size for IN Endpoint Register (USB2D0_INMAXP)

Bit	Mnemonic	Description	Comments
0:2		Reserved	
3	3TRS	3 Multiplier	For Isochronous Endpoints operating in High-Speed mode and with the High-Bandwidth option enabled, the multiplier may only be either two or three (corresponding to 2TRS set or 3TRS set, respectively) and it specifies the maximum number of transactions that can take place in a single microframe. If either 3TRS or 2TRS is non-zero, the function will automatically *split any data packet written to the FIFO into up to two or three USB packets, each containing the specified payload (or less)*. The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be *transmitted* in each microframe. (For Isochronous transfers in Full-Speed mode or if High-Bandwidth is not enabled, bits 3TRS and 2TRS are ignored.)
4	2TRS	2 Multiplier	
5:15	MAXTRS	Maximum Payload/Transaction	Specifies the maximum payload in bytes that can be transferred in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full-Speed and High-Speed operations.

The maximum payload set in this register (multiplied by *m* where *m* = 2 or 3 in the case of High-Bandwidth Isochronous transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated Endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch can cause unexpected results.

The total amount of data represented by the value written to this register (specified payload $\times$ *m*) must not exceed the FIFO size for the IN Endpoint, and should not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the Endpoint, the IN Endpoint FIFO should be completely flushed using the Flush FIFO bit in USB2D0_INCSR after writing the new value to this register.

32.7.5.4 Control/Status for OUT Endpoint Register (USB2D0_OUTCSR)

USB2D0_OUTCSR provides control and status bits for OUT transactions through the currently selected Endpoint. There is an USB2D0_OUTCSR register for each OUT Endpoint (except Endpoint 0).

Address: 0x0 E0000114

Size (bits): 16

Access: R/W

Reset: 0x00

Figure 32-67. Control/Status for OUT Endpoint Register (USB2D0_OUTCSR)

Bit	Mnemonic	Description	Comments
0	AUTOC	Automatically Clear 0 Do not automatically clear the OPR 1 Automatically clear the OPR	If the processor sets this bit, OPR is automatically cleared when the maximum packet size (value in USB2D0_OUTMAXP) is unloaded from the OUT FIFO. If a packet of less than the maximum packet size is unloaded, OPR must be cleared manually. **Note:** This bit should not be set for High-Bandwidth Isochronous Endpoints.
1	ISO	ISO Mode 0 Disable the OUT Endpoint for Bulk/Interrupt transfers 1 Enable the OUT Endpoint for Isochronous transfers	The processor sets this bit to enable the OUT Endpoint for Isochronous transfers, and clears it to enable the OUT Endpoint for Bulk/Interrupt transfers.
2	DMAE	DMA Enable 0 Disable the DMA request for the OUT Endpoint 1 Enable the DMA request for the OUT Endpoint	The processor sets this bit to enable the DMA request for the OUT Endpoint.
3	DN	Disable NYET 0 Enable sending of NYET handshakes 1 Disable sending of NYET handshakes	The processor sets this bit to disable sending of NYET handshakes. When set, all successfully received OUT packets are ACK'd including at the point at which the FIFO becomes full. **Note:** This bit only has any effect in High-speed mode, in which mode it should be set for all Interrupt Endpoints.
4	DMAM	DMA Mode 0 Select DMA Mode 0 1 Select DMA Mode 1	Two modes of operation are supported: DMA Mode 0 in which a DMA request is generated for all received packets, together with an interrupt (if enabled); and DMA Mode 1 in which a DMA request (but no interrupt) is generated for OUT packets of the size specified in USB2D0_OUTMAXP and an interrupt (but no DMA request) is generated for OUT packets of any other size. The MCU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
5:6		Reserved	
7	IRX	Incomplete Receive Operation 0 OPR is cleared 1 Packet in the OUT FIFO is incomplete	This bit is set in a High-Bandwidth Isochronous transfer if the packet in the OUT FIFO is incomplete because parts of the data were not received. It is cleared when OPR is cleared. **Note:** In anything other than a High-Bandwidth Isochronous transfer, this bit will always return 0.

Figure 32-67. Control/Status for OUT Endpoint Register (USB2D0_OUTCSR) (continued)

Bit	Mnemonic	Description	Comments
8	CDT	Clear Data Toggle 0 Do not reset the OUT Endpoint data toggle 1 Reset the OUT Endpoint data toggle	The processor writes 1 to this bit to reset the OUT Endpoint data toggle.
9	SS0	Sent Stall 0 STALL terminated 1 STALL handshake has been transmitted	This bit is set when a STALL handshake has been transmitted. The processor should clear this bit.
10	SS1	Send Stall 0 STALL handshake transmitted 1 Transmit a STALL handshake to a DATA packet	The processor writes 1 to this bit to issue a STALL handshake to a DATA packet. The processor clears this bit to terminate the stall condition. **Note:** This bit has no effect where the Endpoint is being used for Isochronous transfers.
11	FFIFO	Flush FIFO 0 Flush complete 1 Flush the next packet to be read from the OUT Endpoint FIFO	The processor writes 1 to this bit to flush the next packet to be read from the OUT Endpoint FIFO. **Note:** If the FIFO contains two packets, FFIFO will need to be set twice to completely clear the FIFO.
12	DATAE	Data Error 0 OPR is reset 1 Data packet has a CRC error	This bit is set at the same time that OPR is set if the data packet has a CRC error. It is cleared when OPR is cleared. **Note:** This bit is only valid when the Endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
13	OVR	Overrun 0 OUT overrun processed 1 OUT data overrun	This bit is set if an OUT packet arrives while FIFOF is set; that is, the OUT packet cannot be loaded into the OUT FIFO. The processor should clear this bit. **Note:** This bit is only valid when the Endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
14	FIFOF	FIFO Full 0 Space available in the OUT FIFO 1 No more space available in the OUT FIFO	This bit is set when no more packets can be loaded into the OUT FIFO.
15	OPR	Out Packet Ready 0 Data packet unloaded from the OUT FIFO 1 Data packet received in the OUT FIFO	This bit is set when a data packet has been received. The processor should clear this bit when the packet has been unloaded from the OUT FIFO. An interrupt is generated (if enabled) when the bit is set.

32.7.5.5 Maximum Packet Size for OUT Endpoint Register (USB2D0_OUTMAXP)

USB2D0_OUTMAXP defines the maximum amount of data that can be transferred through the selected OUT Endpoint in a single frame/microframe (High-Speed transfers). There is an USB2D0_OUTMAXP register for each OUT Endpoint (except Endpoint 0).

Address: 0x0 E0000116

Size (bits): 16

Access: R/W

Reset: 0x00

Figure 32-68. Maximum Packet Size for OUT Endpoint Register (USB2D0_OUTMAXP)

Bit	Mnemonic	Description	Comments
0:2		Reserved	
3	3TRS	3 Multiplier	For Isochronous Endpoints operating in High-Speed mode and with the High-Bandwidth option enabled, the multiplier may only be either two or three (corresponding to 2TRS set or 3TRS set, respectively) and it specifies the maximum number of transactions that can take place in a single microframe. If either 3TRS or 2TRS is non-zero, the function will automatically *combine the separate USB packets received in any microframe into a single packet within the OUT FIFO*. The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be *received* in each microframe. (For Isochronous transfers in Full-Speed mode or if High-Bandwidth is not enabled, bits 3TRS and 2TRS are ignored.)
4	2TRS	2 Multiplier	
5:15	MAXTRS	Maximum Payload/Transaction	The maximum payload in bytes that can be transferred a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full-Speed and High-Speed operations.

The maximum payload set in this register (multiplied by *m* where *m* = 2 or 3 in the case of High-Bandwidth Isochronous transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated Endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch can cause unexpected results.

The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the OUT Endpoint, and should not exceed half the FIFO size if double-buffering is required.

32.7.5.6 Number of Received Bytes in Endpoint 0 FIFO Register (USB2D0_OUTCOUNT0)

USB2D0_OUTCOUNT0 indicates the number of received data bytes in the Endpoint 0 FIFO.

Note: The value returned changes as the contents of the FIFO change and is only valid while OPR in USB2D0_OUTCSR is set..

Address: 0x0 E000011A (0x0 E000001B when the Index register is set to 0)

Size (bits): 8

Access: R/W

Reset: 0x00

Figure 32-69. Number of Received Bytes in Endpoint 0 FIFO Register (USB2D0_OUTCOUNT0)

Bit	Mnemonic	Description	Comments
0		Reserved	
1:7	CNT	Endpoint 0 OUT Count	Bit 1 is the msb.

32.7.5.7 Number of Bytes in OUT Endpoint FIFO Register (USB2D0_OUTCOUNT)

USB2D0_OUTCOUNT holds the number of received data bytes in the packet in the OUT FIFO.

Note: The value returned changes as the FIFO is unloaded and is only valid while OPR in USB2D0_OUTCSR is set. Also pipeline delays in the CPU/OPB interface may result in the count not being synchronized with the corresponding OPB transaction.

Address: 0x0 E000011A

Size (bits): 16

Access: Read only

Reset: 0x00

Figure 32-70. Number of Bytes in OUT Endpoint FIFO Register (USB2D0_OUTCOUNT)

Bit	Mnemonic	Description	Comments
0:2		Reserved	Always 0.
3:15	CNT	Endpoint OUT count	Bit 3 is the msb.

32.7.6 FIFOs

The function provides four addresses for the processor to access the FIFOs for each Endpoint. Writing to these addresses loads data into the IN FIFO for the corresponding Endpoint. Reading from these addresses unloads data from the OUT FIFO for the corresponding Endpoint. The FIFOs are located on 32-bit full-word boundaries. The address assignment is defined in *Table 32-7*.

Note: Transfers to and from FIFOs may be 8-, 16-, or 32-bit as required, and any combination of access is allowed provided the data accessed is contiguous. However, all the transfers associated with one packet must be of the same width so that the data is consistently byte-, half-word- or full-word-aligned. The last transfer may however contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-half-word transfer.

Table 32-7. FIFO Address Assignment

Endpoint	FIFO Address
0	0x0 E0000120
1	0x0 E0000124
2	0x0 E0000128
3	0x0 E000012C

Part V. Reference

33. Instruction Set

With the exception of the FPU, all instruction processing in the PPC440EPx/GRx is handled by the PPC440 processor. Refer to the *PPC440 Processor User's Manual* for details.

34. Floating Point Instruction Set

Descriptions of the PPC440EPx floating point instructions follow. Each description contains the following elements:

- Instruction names (mnemonic and full)
- Instruction syntax
- Instruction format diagram
- Pseudocode description
- Prose description
- Registers altered

Where appropriate, instruction descriptions list exceptions and invalid instruction forms, and provide programming notes.

Table 34-1 summarizes the PPC440EPx instruction set by category.

Table 34-1. Instruction Categories

Category	Subcategory	Instruction Types
Floating Point	Computational	Elementary arithmetic, multiply-add, rounding and converting, square root and reciprocal estimate
	Noncomputational	Load/store, move, compare, Floating-Point Status and Control Register

34.1 Instruction Set Portability

To support embedded real-time applications, the PPC440EPx implements the defined floating point instruction set of the Book-E Enhanced PowerPC Architecture, with the exception the **fctid**, **fsqrt**, and **fsqrts** instruction.

The Book-E Enhanced PowerPC Architecture defines some instructions that have record forms, often called "dot forms,' that update the CR1 field of the Condition Register (CR). The record forms of these instructions are not implemented.

34.2 Instruction Formats

For more detailed information about instruction formats, including a summary of instruction field usage and instruction format diagrams for the PPC440EPx, see *Floating Point Instruction Summary* on page 973.

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions also have an extended opcode field. The remaining instruction bits are contained in additional fields. All instruction fields belong to one of the following categories:

- Defined

  These instruction fields contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

These fields contain operands, such as general purpose register specifiers and immediate values, each of which may contain any one of a number of values. The instruction format diagrams specify the field names of variable fields.

- Reserved

  Bits in a reserved field should be set to 0. In the instruction format diagrams, reserved fields are shaded.

If any bit in a defined field does not contain the specified value, the instruction is illegal and an Illegal Instruction exception type Program interrupt occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid and its result is architecturally undefined. Unless otherwise noted, the PPC440EPx executes all invalid instruction forms without causing an Illegal Instruction exception.

34.3 Pseudocode

The pseudocode that appears in the instruction descriptions provides a semi-formal language for describing instruction operations.

The pseudocode uses the following notation:

+	Twos complement addition
%	Remainder of an integer division; (33 % 32) = 1.
$\overset{u}{<}$, $\overset{u}{>}$	Unsigned comparison relations
(FPR(r))	The contents of FPR r, where $0 \le r \le 31$.
(FRx)	The contents of an FPR, where *X* is A, B, C, S, or T
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
(Rx)	The contents of a GPR, where *X* is A, B, S, or T
0bn	A binary number
0xn	A hexadecimal number
<, >	Signed comparison relations
=	Assignment
=, ≠	Equal, not equal relations
CEIL(x)	Least integer ≥ x.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies an location in main storage.
EXTS(x)	The result of extending *X* on the left with sign bits.
FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
FRx	An FPR, where *X* is A, B, C, S, or T
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.

GPRs	RA, RB, . . .
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
PC	Program counter.
Rx	A GPR, where *x* is A, B, S, or T
REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
REG[FLD]	A field in a named register
REG_b	A bit in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
$REG_{b:b}$	A range of bits in a named register
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
SPR(SPRN)	A Special Purpose Register (SPR) specified by the SPRF field in an **mfspr** or **mtspr** instruction
$c_{0:3}$	A four-bit object used to store condition results in compare instructions.
do	Do loop. "to" and "by" clauses specify incrementing an iteration variable; "while" and "until" clauses specify terminating conditions. Indenting indicates the scope of a loop.
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.
leave	Leave innermost do loop or do loop specified in a leave statement.
n	A decimal number
^nb	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
||	Concatenation
×	Multiplication
÷	Division yielding a quotient
⊕	Exclusive-OR (XOR) logical operator
–	Twos complement subtraction, unary minus
¬	NOT logical operator
∧	AND logical operator
∨	OR logical operator

34.3.1 Operator Precedence

Table 34-2 lists the pseudocode operators and their associativity in descending order of precedence:

Table 34-2. Operator Precedence

Operators	Associativity
REG_b, REG[FLD], function evaluation	Left to right
n_b	Right to left
¬, – (unary minus)	Right to left
×, ÷	Left to right
+, –	Left to right
||	Left to right
=, ≠, <, >, $\overset{u}{<}$, $\overset{u}{>}$	Left to right
∧, ⊕	Left to right
∨	Left to right
←	None

34.4 Register Usage

Each instruction description lists the registers altered by the instruction. Some register changes are explicitly detailed in the instruction description (for example, the target register of a load instruction). Some instructions also change other registers, but the details of the changes are not included in the instruction descriptions. Common examples of these kinds of register changes include the Floating-Point Registers (FPRs) and the Floating-Point Status and Control Register (FPSCR). For discussion of the FPRs, see *Floating-Point Registers (FPR0:FPR31)* on page 142. For discussion of the FPSCR, see *Floating-Point Status and Control Register (FPSCR)* on page 142.

34.5 Floating-Point Instructions

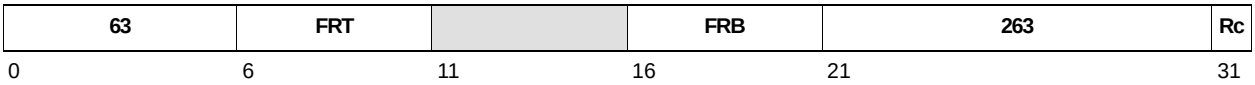
Primary opcode 63 is used for the double-precision arithmetic instructions and miscellaneous instructions (for example, the *Floating-Point Status and Control Register Manipulation* instructions). Primary opcode 59 is used for the single-precision arithmetic instructions.

The single-precision instructions for which there is a corresponding double-precision instruction have the same format and extended opcode as that double-precision instruction.

34.6 Alphabetical Instruction Listing

The following pages list the defined floating point instructions implemented in the PPC440EPx.

fabs FRT, FRB Rc = 0



(FRT) ← 0 || $(FRB)_{1:63}$

The contents of FRB, with bit 0 set to zero, are placed into FRT.

Registers Altered

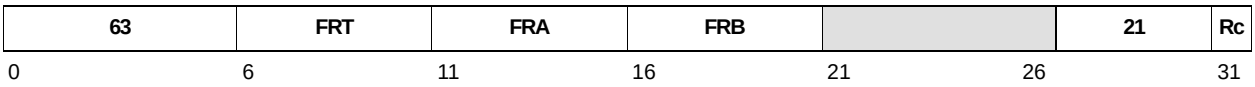
- FRT

Exceptions

An attempt to execute **fabs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fadd FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) + (FRB)

The floating-point operand in FRA is added to the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN] and placed into FRT.

Floating-point addition is based on exponent comparison and addition of the two significands. The exponents of the two operands are compared, and the significand accompanying the smaller exponent is shifted right, with its exponent increased by one for each bit shifted, until the two exponents are equal. The two significands are then added or subtracted as appropriate, depending on the signs of the operands, to form an intermediate sum. All 53 bits of the significand and the three guard bits (G, R, X) enter into the computation.

If a carry occurs, the significand of the sum is shifted right one bit position and the exponent is increased by one.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

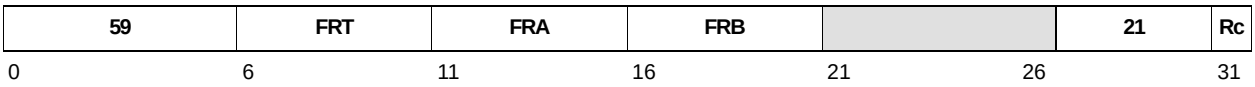
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]

Exceptions

An attempt to execute **fadd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fadds FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) $+_{sp}$ (FRB)

The floating-point operand in FRA is added to the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN] and placed into FRT.

Floating-point addition is based on exponent comparison and addition of the two significands. The exponents of the two operands are compared, and the significand accompanying the smaller exponent is shifted right, with its exponent increased by one for each bit shifted, until the two exponents are equal. The two significands are then added or subtracted as appropriate, depending on the signs of the operands, to form an intermediate sum. All 53 bits of the significand and the three guard bits (G, R, X) enter into the computation.

If a carry occurs, the significand of the sum is shifted right one bit position and the exponent is increased by one.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

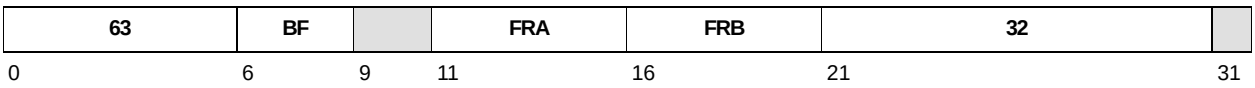
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]

Exceptions

An attempt to execute **fadds** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fcmpo BF, FRA, FRB



if (FRA) is a NaN or
(FRB) is a NaN then c ← 0b0001
else if (FRA) < (FRB) then c ← 0b1000
else if (FRA) > (FRB) then c ← 0b0100
else c← 0b0010
FPSCR[FPCC] ← c
$CR_{4 \times BF:4 \times BF + 3}$ ← c
if (FRA) is a SNaN or (FRB) is a SNaN then do
  if FPSCR[VE] = 0 then FPSCR[VXVC] ← 1
else if (FRA) is a QNaN or (FRB) is a QNaN then FPSCR[VXVC] ← 1

The floating-point operand in FRA is compared to the floating-point operand in FRB. The result of the compare is placed into the CR field BF and FPSCR[FPCC]. The comparison ignores the sign of zero (that is, +0 is considered equal to –0).

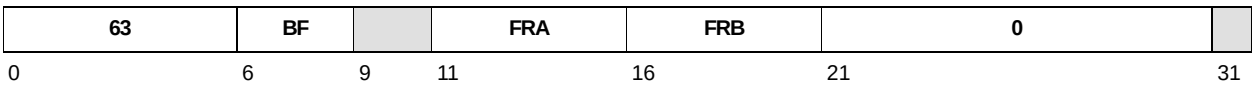
If (FRA) or (FRB) is a NaN, either quiet or signaling, the CR field BF and FPSCR[FPCC] are set to reflect unordered.

If (FRA) or (FRB) is a Signaling NaN and Invalid Operation is disabled (FPSCR[VE] = 0), FPSCR[VXVC] = 1. If neither (FRA) nor (FRB) is a Signaling NaN, but at least one operand is a Quiet NaN, FPSCR[VXVC] = 1.

Registers Altered

- CR[BF]
- FPSCR[FPCC, FX, VXSNAN, VXVC]

fcmpu BF, FRA, FRB



if (FRA) is a NaN or
(FRB) is a NaN then c ← 0b0001
else if (FRA) < (FRB) then c ← 0b1000
else if (FRA) > (FRB) then c ← 0b0100
else c← 0b0010
FPSCR[FPCC] ← c
$CR_{4 \times BF:4 \times BF+3}$ ← c
if ((FRA) is a SNaN or (FRB) is a SNaN) then FPSCR[VXSNAN] ← 1

The floating-point operand in FRA is compared to the floating-point operand in FRB. The result of the compare is placed into the CR field BF and FPSCR[FPCC]. The comparison ignores the sign of zero (that is, +0 is considered equal to –0).

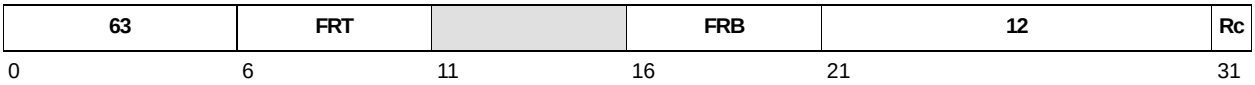
If (FRA) or (FRB) is a NaN, either quiet or signaling, the CR field BF and FPSCR[FPCC] are set to reflect unordered.

If either (FRA) or (FRB) is a Signaling NaN, FPSCR[VXSNAN] = 1.

Registers Altered

- CR
- FPSCR[FPCC, FX, VXSNAN]

fctiw FRT, FRB Rc = 0



The floating-point operand in FRB is converted to a 32-bit signed integer, using the rounding mode specified by FPSCR[RN], and placed into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.

If (FRB) is greater than 2^{31}–1, $FRT_{32:63}$ are set to 0x7FFFFFFF. If (FRB) is less than -2^{31}, $FRT_{32:63}$ are set to 0x80000000.

Except for enabled Invalid Operation Exceptions, FPSCR[FPRF] is undefined. FPSCR[FR] is set if the result is incremented when rounded. FPSCR[FI] is set if the result is inexact.

Registers Altered

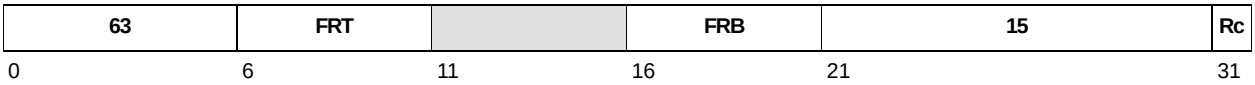
- FRT
- FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI]
- FPSCR[FPRF] undefined

Exceptions

An attempt to execute **fctiw** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fctiwz FRT, FRB Rc = 0



The floating-point operand in FRB is converted to a 32-bit signed integer, using the *Round toward Zero* rounding mode, and placed into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.

If (FRB) is greater than 2^{31}–1, $FRT_{32:63}$ are set to 0x7FFFFFFF. If (FRB) is less than -2^{31}, $FRT_{32:63}$ are set to 0x80000000.

Except for enabled Invalid Operation Exceptions, FPSCR[FPRF] is undefined. FPSCR[FR] is set if the result is incremented when rounded. FPSCR[FI] is set if the result is inexact.

Registers Altered

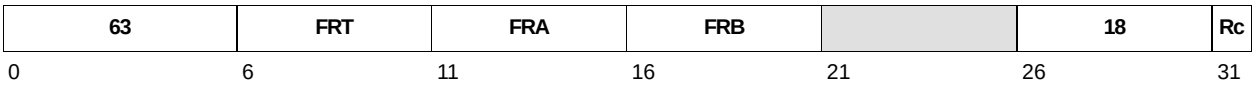
- FRT
- FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI]
- FPSCR[FPRF] undefined

Exceptions

An attempt to execute **fctiwz** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fdiv FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) ÷ (FRB)

The floating-point operand in FRA is divided by the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

The remainder is not supplied as a result.

Floating-point division is based on exponent subtraction and division of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

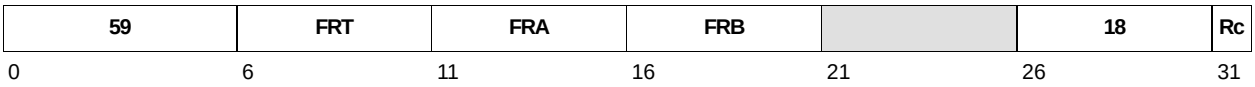
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]

Exceptions

An attempt to execute **fdiv** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fdivs FRT, FRA, FRB Rc = 0



FRT ← (FRA) ÷ (FRB)

The floating-point operand in FRA is divided by the floating-point operand in FRB.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into (FRT).

The remainder is not supplied as a result.

Floating-point division is based on exponent subtraction and division of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

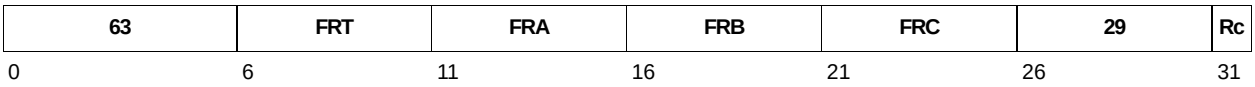
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]

Exceptions

An attempt to execute **fdivs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmadd FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] + (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

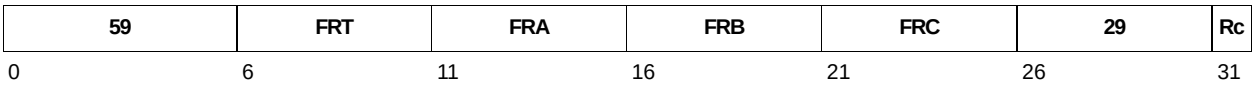
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmadd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmadds FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] + (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

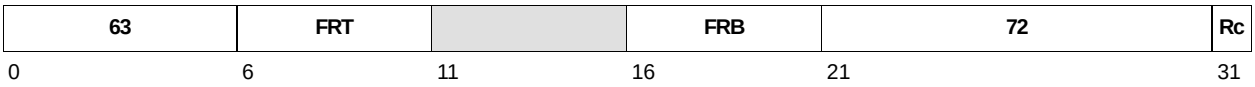
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmadds** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmr FRT, FRB Rc = 0



(FRT) ← FR(FRB)

The contents of FRB are placed into FRT.

Registers Altered

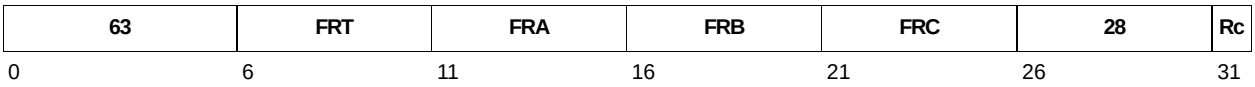
- FRT

Exceptions

An attempt to execute **fmr** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmsub FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] – (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

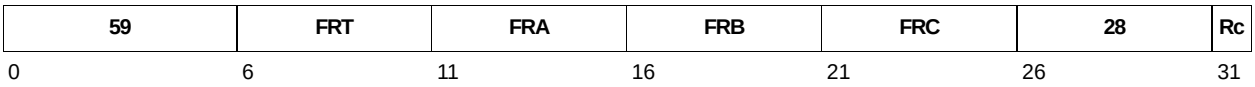
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmsub** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmsubs FRT, FRA, FRC, FRB Rc = 0



(FRT) ← [(FRA) × (FRC)] – (FRB)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation exceptions when FPSCR[VE] = 1.

Registers Altered

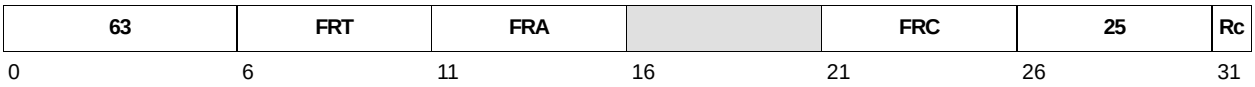
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fmsubs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmul FRT, FRA, FRC Rc = 0



(FRT) ← (FRA) × (FRC)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

Floating-point multiplication is based on exponent addition and multiplication of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation exceptions when FPSCR[VE] = 1.

Registers Altered

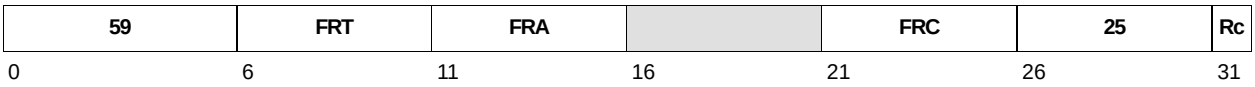
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]

Exceptions

An attempt to execute **fmul** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fmuls FRT, FRA, FRC Rc = 0



(FRT) ← (FRA) × (FRC)

The floating-point operand in FRA is multiplied by the floating-point operand in FRC.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into FRT.

Floating-point multiplication is based on exponent addition and multiplication of the significands.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation exceptions when FPSCR[VE] = 1.

Registers Altered

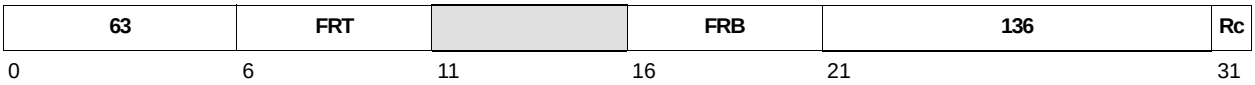
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]

Exceptions

An attempt to execute **fmuls** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnabs FRT, FRB Rc = 0



$(FRT) \leftarrow 1 \,||\, (FRB)_{1:63}$

The contents of FRB, with bit 0 set to one, are placed into FRT.

Registers Altered

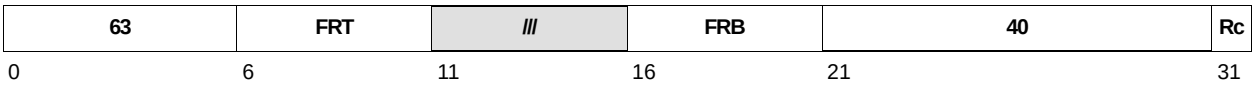
- FRT

Exceptions

An attempt to execute **fnabs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fneg FRT, FRB Rc = 0



$(FRT) \leftarrow \neg(FRB)_0 \,||\, (FRB)_{1:63}$

The contents of FRB, with bit 0 inverted, are placed into FRT.

Registers Altered

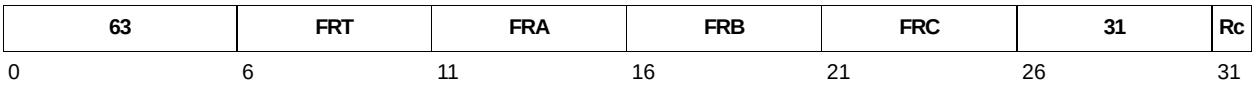
- FRT

Exceptions

An attempt to execute **fneg** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmadd FRT, FRA, FRC, FRB Rc = 0



FRT ← –([(FRA) × (FRC)] + (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN], then negated and placed into (FRT).

This instruction produces the same result as would be obtained by using the **fmadd** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their "sign" (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a "sign" bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the "sign" bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

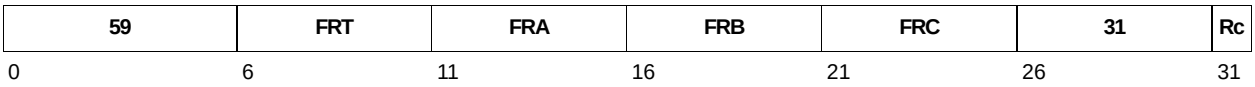
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmadd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmadds FRT, FRA, FRC, FRB Rc = 0



(FRT) ← –([(FRA) × (FRC)] + (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is added to this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN], then negated and placed into FRT.

This instruction produces the same result as would be obtained by using the **fmadds** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their "sign" (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a "sign" bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the "sign" bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

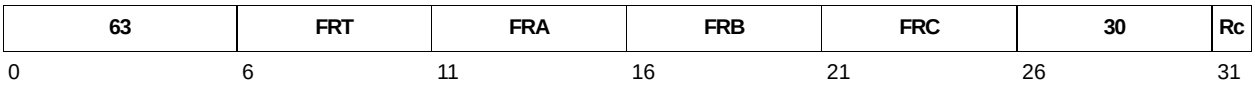
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmadds** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmsub FRT, FRA, FRC, FRB Rc = 0



(FRT) ← –([(FRA) × (FRC)] – (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of the FPSCR[RN], then negated and placed into FRT.

This instruction produces the same result as would be obtained by using the **fmsub** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their ”sign” (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a “sign” bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the “sign” bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

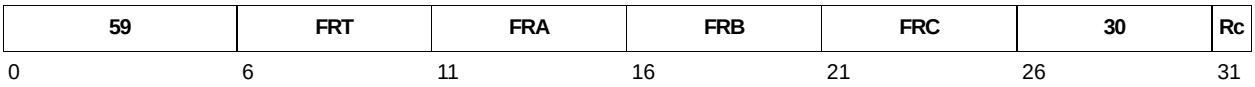
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmsub** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fnmsubs FRT, FRA, FRC, FRB Rc = 0



(FRT) ← –([(FRA) × (FRC)] – (FRB))

The floating-point operand in FRA is multiplied by the floating-point operand in FRC. The floating-point operand in FRB is subtracted from this intermediate result.

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of the FPSCR[RN], then negated and placed into FRT.

This instruction produces the same result as would be obtained by using the **fmsubs** instruction and then negating the result, with the following exceptions.

- Quiet NaNs propagate with no effect on their ”sign” (high-order) bit.
- Quiet NaNs that are generated as the result of a disabled Invalid Operation Exception have a “sign” bit of 0.
- Signaling NaNs that are converted to Quiet NaNs as the result of a disabled Invalid Operation Exception retain the “sign” bit of the Signaling NaN.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

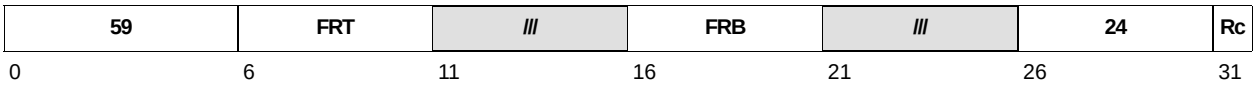
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]

Exceptions

An attempt to execute **fnmsubs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fres FRT, FRB (Rc=0)



(FRT) ← FPReciprocalEstimate(FRB)

A single-precision estimate of the reciprocal of the floating-point operand in FRB is placed into FRT. The estimate is correct to a precision of one part in 2^{13} of the reciprocal of (FRB), that is,

$$\left| \frac{\text{estimate} - \frac{1}{x}}{\frac{1}{x}} \right| \leq \frac{1}{2}13$$

where *x* is the initial value of (FRB).

Table 34-3 summarizes operation with operands having various special values.

*Table 34-3. **fres** Operation with Special Operand Values*

Operand	Result	Exception
–∞	–0	—
–0	–∞[1]	ZX
+0	+∞[1]	ZX
+∞	+0	—
SNaN	QNaN[2]	VXSNAN
QNaN	QNaN	—

Note: 1. No result if FPSCR[ZE] = 1
2. No result if FPSCR[VE] = 1.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

- FRT
- FPSCR[FX, OX, UX, ZX, VXSNAN]
- FPSCR[FPRF, FR, FI] undefined

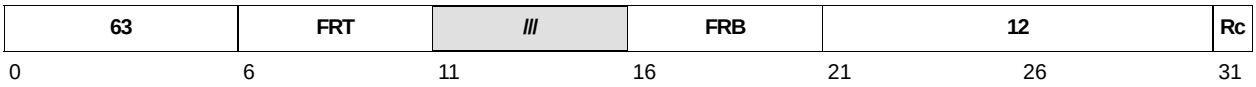
Exceptions

An attempt to execute **fres** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Architecture Notes

The value placed into FRT may vary between implementations, and between different executions on the same implementation.

frsp FRT, FRB (Rc=0)



The floating-point operand in FRB is rounded to single-precision, using the rounding mode specified by FPSCR[RN], and placed into FRT.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

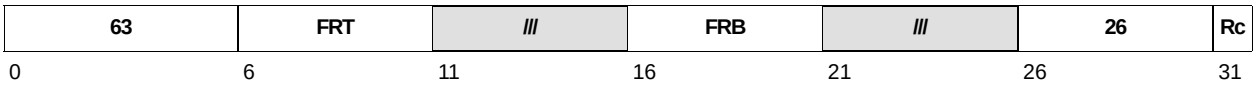
- FRT
- FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN]

Exceptions

An attempt to execute **frsp** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

frsqrte FRT, FRB (Rc=0)



(FRT) ← FPReciprocalSquareRootEstimate(FRB)

A double-precision estimate of the reciprocal of the square root of the floating-point operand in FRB is placed into FRT. The estimate placed is correct to a precision of one part in 2^{13} of the reciprocal of the square root of (FRB), that is,

$$\left| \frac{\left(\text{estimate} - \frac{1}{\sqrt{x}}\right)}{\frac{1}{\sqrt{x}}} \right| \leq \frac{1}{2}13$$

where *x* is the initial value in FPR(FRB). Note that the value placed into FPR(FRT) may vary between implementations, and between different executions on the same implementation.

Table 34-4 summarizes operation with various special values of the operand.

*Table 34-4. **frsqrte** Operation with Special Operand Values*

Operand	Result	Exception
–∞	QNaN[2]	VXSQRT
< 0	QNaN[2]	VXSQRT
–0	–∞[1]	ZX
+0	+∞[1]	ZX
+∞	+0	—
SNaN	QNaN[2]	VXSNAN
QNaN	QNaN	—

Note: 1. No result if FPSCR[ZE] = 1
2. No result if FPSCR[VE] = 1.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1 and Zero Divide Exceptions when FPSCR[ZE] = 1.

Registers Altered

- FRT
- FPSCR[FX, ZX, VXSNAN, VXSQRT]
- FPSCR[FPRF, FR, FI] undefined

Exceptions

An attempt to execute **frsqrte** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

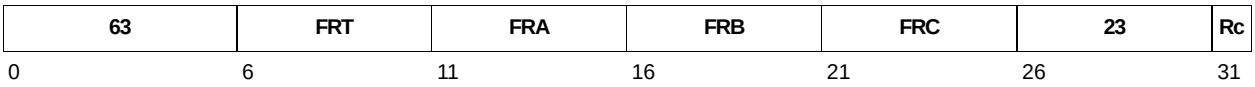
Architecture Notes

No single-precision version of this instruction is provided. If the floating-point operand in FRB is representable in single format, so is the resulting value of FRT.

Implementation Note

The precision of this estimate in this implementation is higher than that specified in Book-E (2^{-5}). Therefore, code relying on this higher precision may not work on other PowerPC implementations.

fsel FRT, FRA, FRC, FRB (Rc=0)



if (FRA) ≥ 0.0 then (FRT) ← (FRC)

else (FRT) ← (FRB)

The floating-point operand in FRA is compared to zero. If the operand is greater than or equal to 0, FRT is set to the contents of FRC. If the operand is less than 0 or is a NaN, FRT is set to the contents of FRB. The comparison ignores the sign of zero (that is, +0 is considered equal to –0).

Registers Altered

- FRT

Exceptions

An attempt to execute fsel[.] while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Programming Notes

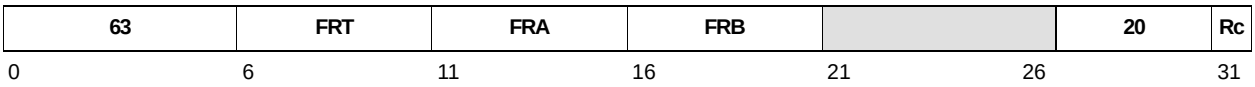
Care must be taken in using fsel if compatibility with IEEE 754 is required, or if the values being tested can be NaNs or infinities. See XREF TBD.

Examples of uses of this instruction can be found in XREF TBD.

Architecture Notes

The *Select* instruction is similar to a *Move* instruction, and therefore does not alter FPSCR.

fsub FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) – (FRB)

The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to double precision under control of FPSCR[RN] and placed into FRT.

The operation of **fsub** is identical to that of **fadd**, except that the contents of FRB participate in the operation with the sign bit (bit 0) inverted.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

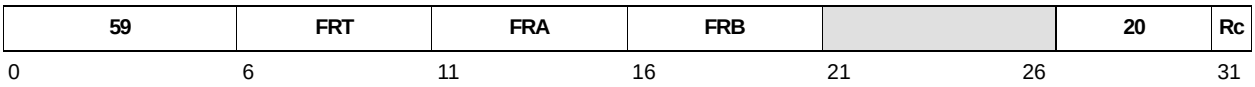
- FRT
- FPSRP[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN]

Exceptions

An attempt to execute **fsub** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

fsubs FRT, FRA, FRB Rc = 0



(FRT) ← (FRA) – (FRB)

The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).

If the most significant bit of the resultant significand is not 1, the result is normalized. The result is rounded to single precision under control of FPSCR[RN] and placed into FRT.

The operation of **fsubs** is identical to that of **fadds**, except that the contents of FRB participate in the operation with the sign bit (bit 0) inverted.

FPSCR[FPRF] is set to the class and sign of the result, except for Invalid Operation Exceptions when FPSCR[VE] = 1.

Registers Altered

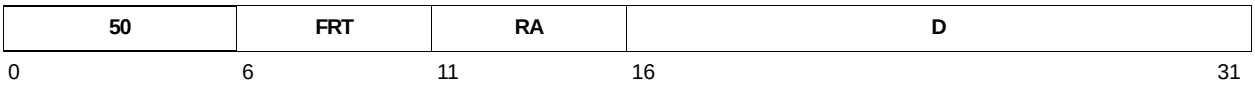
- FRT
- FPSCR[FPRF, FR, FI FX, OX, UX, XX, VXSNAN]

Exceptions

An attempt to execute **fsubs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

lfd FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← MEM(EA,8)

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The doubleword in storage addressed by EA is placed into FRT.

Registers Altered

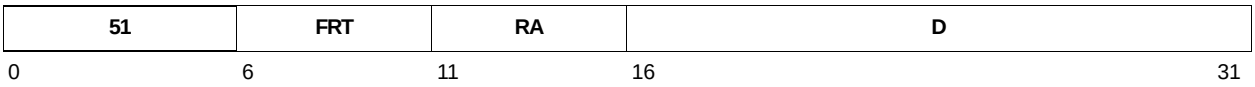
- FRT

Exceptions

An attempt to execute **lfd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

lfdu FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← MEM(EA,8)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The doubleword in storage addressed by EA is placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

Exceptions

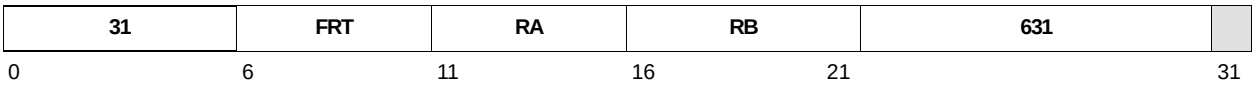
An attempt to execute **lfdu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

Invalid Forms

RA = 0

lfdux FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← MEM(EA,8)
RA ← EA

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The doubleword in storage addressed by EA is placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

Exceptions

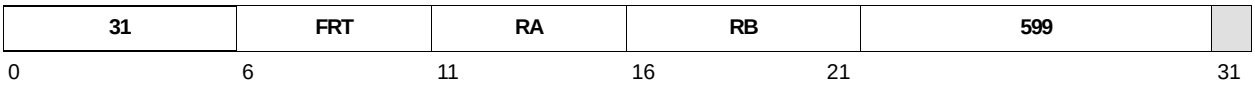
An attempt to execute **lfdux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

Invalid Forms

RA = 0

lfdx FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← MEM(EA,8)

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The doubleword in storage addressed by EA is placed into FRT.

Registers Altered

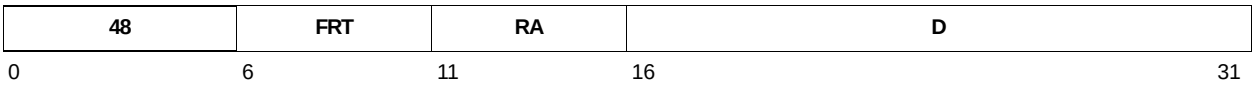
- FRT

Exceptions

An attempt to execute **lfdx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

lfs FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← DOUBLE(MEM(EA,4))

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 149) and placed into FRT.

Registers Altered

- FRT

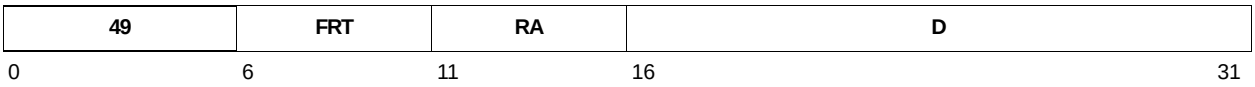
Exceptions

An attempt to execute **lfs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfsu FRT, D(RA)



EA ← (RA|0) + EXTS(D)
(FRT) ← DOUBLE(MEM(EA,4))
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D field to 32 bits.

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 149) and placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

Exceptions

An attempt to execute **lfsu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

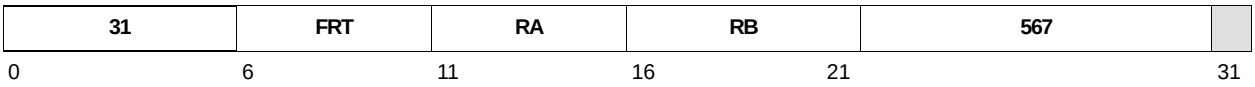
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfsux FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← DOUBLE(MEM(EA,4))
RA ← EA

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 149) and placed into FRT.

EA is placed into RA.

Registers Altered

- FRT
- RA

Exceptions

An attempt to execute **lfsux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

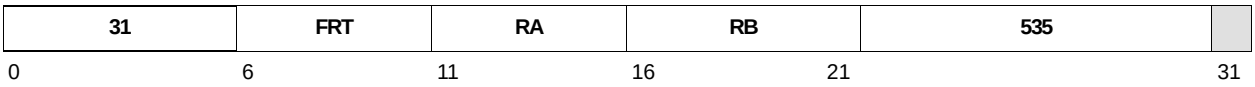
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

lfsx FRT, RA, RB



EA ← (RA|0) + (RB)
(FRT) ← DOUBLE(MEM(EA,4))

Let the effective address (EA) be the sum (RA|0) and the index specified by (RB).

The word in storage addressed by EA is interpreted as a floating-point single-precision operand. This word is converted to floating-point double format (see *Data Handling and Precision* on page 149) and placed into FRT.

Registers Altered

- FRT

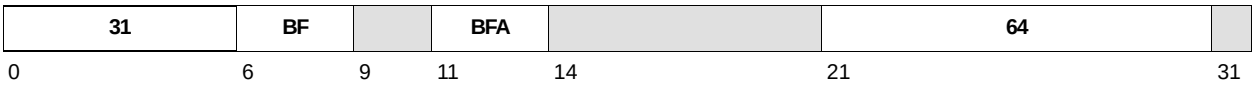
Exceptions

An attempt to execute **lfsx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

mcrfs BF, BFA



$CR_{BF \times 4:BF \times 4+3} \leftarrow FPSCR_{BFA \times 4:BFA \times 4 + 3}$

$FPSCR_{BFA \times 4:BFA \times 4 + 3} \leftarrow 0b0000$

The contents of the FPSCR specified by BFA are copied to the CR field specified by BF. All exception bits that are copied are set to 0 in the FPSCR. If FPSCR[FX] is copied, it is set to 0 in the FPSCR.

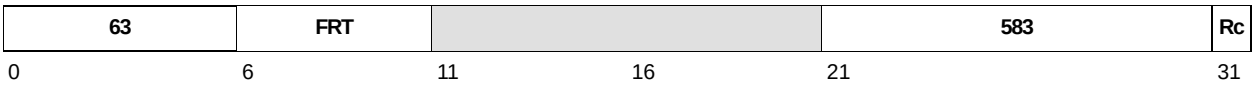
Registers Altered

- CR field BF
- FPSCR[FX, OX] if BFA=0
- FPSCR[UX, ZX, XX, VXSNAN] if BFA=1
- FPSCR[VXISI, VXIDI, VXZDZ, VXIMZ] if BFA=2
- FPSCR[VXVC] if BFA=3
- FPSCR[VXSOFT, VXSQRT, VXCVI] if BFA=5

Exceptions

An attempt to execute **mcrfs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

mffs FRT Rc = 0



(FRT) ← FPSCR

The contents of the FPSCR are placed into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.

Registers Altered

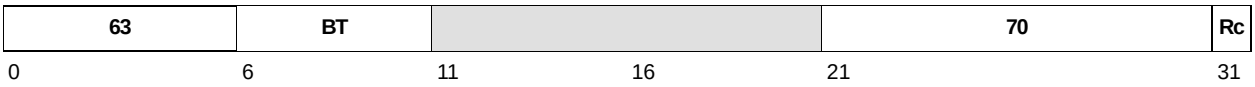
- FRT

Exceptions

An attempt to execute **mffs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

mtfsb0 BT Rc = 0



$FPSCR_{BT} \leftarrow 0$

Bit BT of the FPSCR is set to 0.

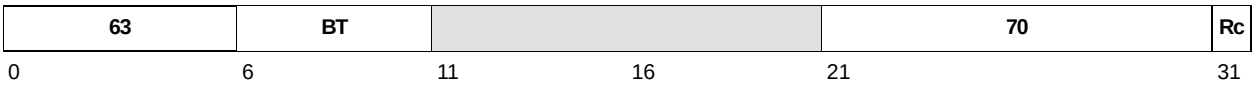
Registers Altered

- FPSCR[BT]

An attempt to execute **mtfsb0** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

mtfsb1 BT Rc = 0



$FPSCR_{BT} \leftarrow 1$

Bit BT of the FPSCR is set to 1.

Registers Altered

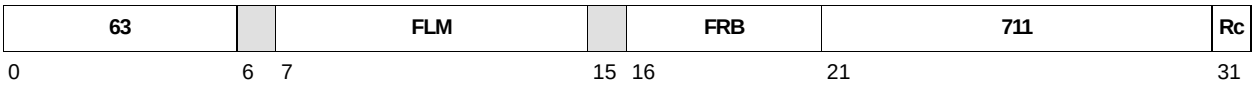
- FPSCR[BT, FX]

AIf Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

Programming Notes

FPSCR[FEX, VX] cannot be explicitly set.

mtfsf FLM, FRB Rc = 0



i ← 0
do while i<8
  if FLM_i=1 then $FPSCR_{4 \times i:4 \times i + 3} \leftarrow (FRB)_{4 \times i:4 \times i + 3}$
  i ← i+1

The contents of bits 32:63 of FRB are placed into the FPSCR under control of the field mask specified by FLM. The field mask identifies the affected 4-bit fields. Let *i* be an integer in the range 0–7. If FLM_i = 1, FPSCR field *i* (FPSCR bits $4 \times i - 4 \times i + 3$) is set to the contents of the corresponding field of the low-order 32 bits of FRB.

FPSCR[FX] is altered only if FLM_0 = 1.

Registers Altered

- FPSCR fields selected by mask

Exceptions

An attempt to execute **mtfsf** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

Programming Notes

Updating fewer than all eight fields of the FPSCR may have substantially poorer performance on some implementations than updating all the fields.

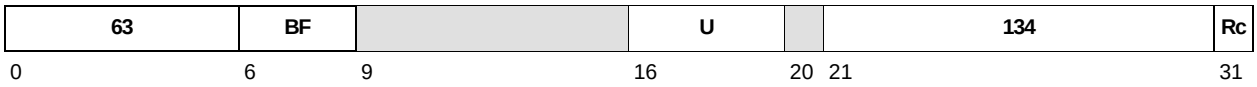
When $FPSCR_{0:3}$ is specified, bits 0 and 3 (FPSCR[FX, OX]) are set to the values of FRB_{32} and FRB_{35}. Even if **mtfsf** causes FPSCR[OX] to change from 0 to 1, FPSCR[FX] is set from FRB_{32}, not by the usual rule that FPSCR[FX] is set to 1 when an exception bit changes from 0 to 1. Bits 1 and 2 (FPSCR[FEX, VX]) are set according to the usual rule, given on XREF TBD, and not from $FRB_{33:34}$.

Implementation Note

In general, the source FPR should contain data produced by a double-precision load operation. The value moved to FPSCR is unpredictable if the source FPR contains denormalized single format data (that is, data that cannot be represented as a normalized single-precision number that was produced from a single precision operation.

This is similar to the achitectural rule for **stfiwx**.

mtfsfi BF, U Rc = 0



$FPSCR_{BF \times 4:BF \times 4 + 3} \leftarrow U$

The value of the U field is placed into FPSCR field BF.

$FPSCR_{FX}$ is altered only if BF = 0.

Registers Altered

- $FPSCR_{BF}$

Exceptions

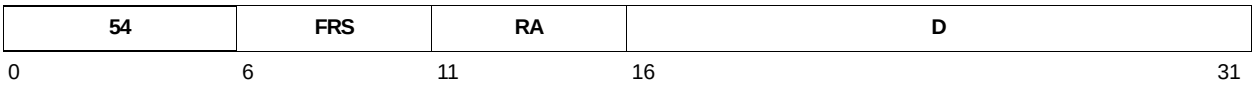
An attempt to execute **mtfsfi** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

If Rc = 1, an Unimplemented Operation exception type Program interrupt occurs.

Programming Notes

When $FPSCR_{0:3}$ is specified, bits 0 and 3 FPSCR[FX, OX] are set to the values of U_0 and U_3. Even if **mtfsfi** causes FPSCR[OX] to change from 0 to 1, FPSCR[FX] is set from U_0 and not by the usual rule that FPSCR[FX] is set to 1 when an exception bit changes from 0 to 1). Bits 1 and 2 (FPSCR[FEX, VX]) are set according to the usual rule, given on XREF TBD, and not from $U_{1:2}$.

stfd FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,8) ← (FRS)

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are stored into the doubleword in storage addressed by EA.

Registers Altered

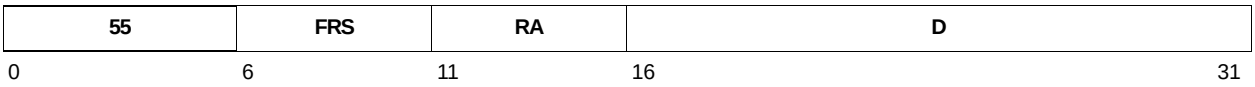
- None

Exceptions

An attempt to execute **stfd** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

stfdu FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,8) ← (FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are stored into the doubleword in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

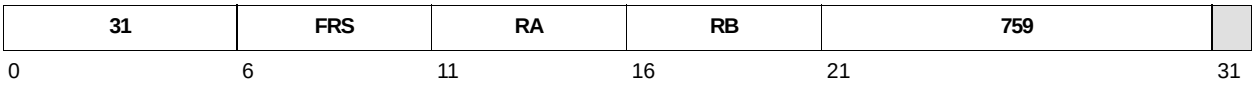
An attempt to execute **stfdu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

Invalid Forms

RA = 0

stfdux FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,8) ← (FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are stored into the doubleword in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

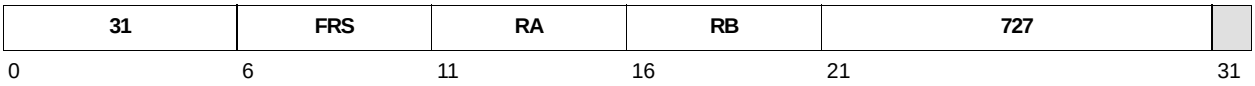
An attempt to execute **stfdux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

Invalid Forms

RA = 0

stfdx FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,8) ← (FRS)

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are stored into the doubleword in storage addressed by EA.

Registers Altered

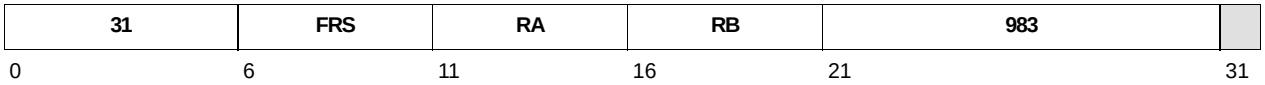
- None

Exceptions

An attempt to execute **stfdx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

An alignment exception occurs if the EA is not aligned on a double word boundary.

stfiwx FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,4) ← $(FRS)_{32:63}$

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of bits 32:63 of FRS are stored, without conversion, into the word in storage addressed by EA.

If the contents of FRS were produced, either directly or indirectly, by a *Load Floating-Point Single* instruction, a single-precision *Arithmetic* instruction, or **frsp**, the value stored is undefined. (The contents of FRS are produced directly by such an instruction if FRS is the target register for the instruction. The contents of FRS are produced indirectly by such an instruction if FRS is the final target register of a sequence of one or more *Floating-Point Move* instructions, with the input to the sequence having been produced directly by such an instruction.)

Registers Altered

- None

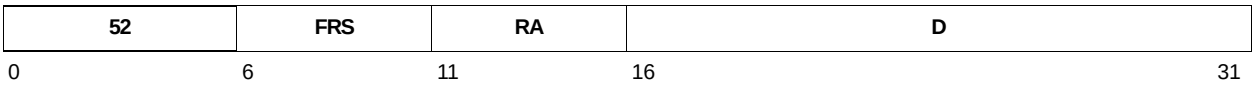
Exceptions

An attempt to execute **stfiwx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfs FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,4) ← SINGLE(FRS)

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are converted to single-precision format (see *Data Handling and Precision* on page 149) and stored into the word in storage addressed by EA.

Registers Altered

- None

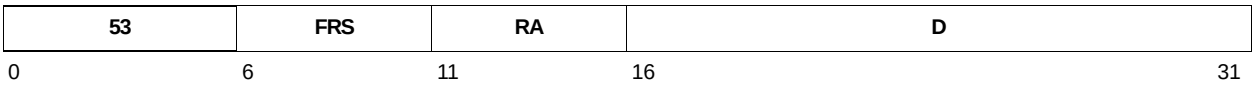
Exceptions

An attempt to execute **stfs** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfsu FRS, D(RA)



EA ← (RA|0) + EXTS(D)
MEM(EA,4) ← SINGLE(FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the displacement obtained by sign-extending the 16-bit D instruction field to 32 bits.

The contents of FRS are converted to single-precision format (see *Data Handling and Precision* on page 149) and stored into the word in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

An attempt to execute **stfsu** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

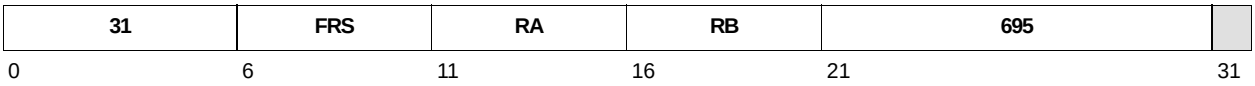
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfdux FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,4) ← SINGLE(FRS)
RA ← EA

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are converted to single-precision format (see *Data Handling and Precision* on page 149) and stored into the word in storage addressed by EA.

EA is placed into RA.

Registers Altered

- RA

Exceptions

An attempt to execute **stfsux** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

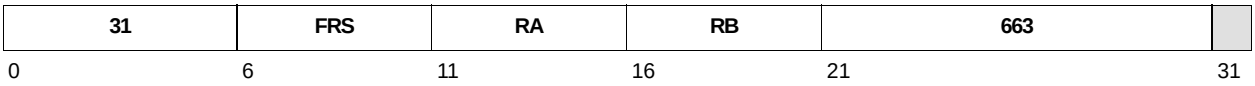
Invalid Forms

RA = 0

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

stfsx FRS, RA, RB



EA ← (RA|0) + (RB)
MEM(EA,4) ← SINGLE(FRS)

Let the effective address (EA) be the sum of (RA|0) and the index specified by (RB).

The contents of FRS are converted to single-precision format (see*Data Handling and Precision* on page 149) and stored into the word in storage addressed by EA.

Registers Altered

- None

Exceptions

An attempt to execute **stfsx** while MSR[FP] = 0 causes a Floating-Point Unavailable exception.

Implementation Note

The EA must be a multiple of 4. Otherwise, an Alignment exception occurs.

35. Register Summary

This chapter provides an alphabetical listing and bit definitions for the registers provided by the PPC440EPx/GRx Embedded Processor that are not associated with the PPC440 processor. These registers include the DCRs, FPRs, MMIO registers, and all registers provided by the peripheral functions.

35.1 Reserved Fields

For all registers with fields marked as reserved, the reserved fields should be written as *zero* and read as *undefined*. That is, when writing to a reserved field, write a zero to that field. When reading from a reserved field, ignore that field.

The recommended coding practice is to perform the initial write to a register with reserved fields as described in the preceding paragraph, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register, alter desired fields with logical instructions, and then write the register.

35.2 Device Control Registers

DCRs may be used to control various on-chip system functions, such as the operation of on-chip buses, peripherals, and certain processor function behaviors. The DCR access instructions are **mtdcr** (move to device control register) and **mfdcr** (move from device control register), which move data between GPRs and the DCRs. Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR.

See *Table 3-3* on page 115 for a complete listing of the PPC440EPx/GRx Device Control registers.

35.3 Memory Mapped Registers

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses.

See *Table 3-4* on page 125 for a complete listing of the PPC440EPx/GRx Memory Mapped registers.

35.4 Alphabetical Listing of Chip Control and Peripheral Function Registers

The following pages list all the device control registers (DCRs), floating-point registers (FPRs), and memory mapped registers (MMIOs) implemented in the PPC440EPx/GRx Embedded Processor For each register, the following information is supplied:

- Register mnemonic
- Register type
- Register description
- Register number or address
- Access (Read/Clear, Read-Only, Read/Write, Write-Only)
- Cross-reference to detailed register information

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers

Register	Type	Description	Address or Number	Access	See Page
CPM0_ER	DCR	CPM 0 Enable Register	0x00B0	R/W	291
CPM0_FR	DCR	CPM 0 Force Register	0x00B1	R/W	293
CPM0_SR	DCR	CPM 0 Status Register	0x00B2	R/W	294
CPM1_ER	DCR	CPM 1 Enable Register	0x00F0	R/W	291
CPM1_FR	DCR	CPM 1 Force Register	0x00F1	R/W	293
CPM1_SR	DCR	CPM 1 Status Register	0x00F2	R/W	294
CPR0_CFGADDR	DCR	Clock, Power-On, Reset Configuration Address Register	0x000C	R/W	284
CPR0_CFGDATA	DCR	Clock, Power-On, Reset Configuration Data Register	0x000D	R/W	284
CPR0_CLKUPD	DCR	Clocking Update Register	offset 0x0020	R/W	285
CPR0_ICFG	DCR	Initial Configuration Register	offset 0x0140	R/W	176
CPR0_MALD	DCR	MAL Clock Divisor Register	offset 0x0100	R/W	289
CPR0_OPBD0	DCR	OPB Clock Divisor Register	offset 0x00C0	R/W	288
CPR0_PERD0	DCR	Peripheral Clock Divisor Register	offset 0x00E0	R/W	288
CPR0_PLLC0	DCR	PLL Control Register	offset 0x0040	R/W	285
CPR0_PLLD0	DCR	PLL Divisor Register	offset 0x0060	R/W	286
CPR0_PRIMAD0	DCR	Primary Divisor Register A	offset 0x0080	R/W	287
CPR0_PRIMBD0	DCR	Primary Divisor Register B	offset 0x00A0	R/W	288
CPR0_SPCID	DCR	Synchronous PCI Clock Divisor Register	offset 0x0120	R/W	289
CRYP0_DC_CTRL	MMIO	Device Control Register	0x0 E016 0080	R/W	378
CRYP0_DC_DEVID	MMIO	Device ID Register	0x0 E016 0084	R	378
CRYP0_DC_DEVINF	MMIO	Device Information Register	0x0 E016 0088	R	379
CRYP0_DMA_CFG	MMIO	DMA Configuration/Status Register	0x0 E016 00D4	R/W	382
CRYP0_DMA_UCMD	MMIO	DMA Command Register	0x0 E016 009C	R/W	381
CRYP0_DMA_UDST	MMIO	DMA Destination Address Register	0x0 E016 0098	R/W	380
CRYP0_DMA_USRC	MMIO	DMA Source Address Register	0x0 E016 0094	R/W	380
CRYP0_INT_CFG	MMIO	Interrupt Configuration Register	0x0 E015 00AC	R/W	376
CRYP0_INT_DESCT	MMIO	Interrupt Descriptor Count Register	0x0 E015 00B4	R/W	377
CRYP0_INT_DESRD	MMIO	Interrupt Force Descriptor Read Register	0x0 E015 00B0	W	377
CRYP0_INT_EN	MMIO	Interrupt Mask Register	0x0 E015 00A8	R/W	375
CRYP0_INT_MSK	MMIO	Interrupt Masked Status Register	0x0 E015 00A4	R/W	374
CRYP0_INT_UNMSK	MMIO	Interrupt Unmasked Status Register	0x0 E015 00A0	R	374
CRYP0_PE_CTLST	MMIO	PE Control/Status Register	0x0 E010 0000	R/W	313
CRYP0_PE_DEST	MMIO	PE Destination Address Register	0x0 E010 0008	R/W	319
CRYP0_PE_DMA_CF	MMIO	PE DMA Configuration Register	0x0 E010 0040	See register	322
CRYP0_PE_DMA_ST	MMIO	PE DMA Status Register	0x0 E010 0044	R	324
CRYP0_PE_E_RING	MMIO	PE External Ring Status Register	0x0 E010 005C	R	331
CRYP0_PE_GATH	MMIO	PE Gather Particle Ring Base Address Register	0x0 E010 0064	R/W	332

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
CRYP0_PE_I_RING	MMIO	PE Internal Ring Status Register	0x0 E010 0058	R	330
CRYP0_PE_IO_THR	MMIO	PE I/O Threshold Register	0x0 E010 0060	R/W	331
CRYP0_PE_LENGTH	MMIO	PE Length Register	0x0 E010 0010	R/W	321
CRYP0_PE_PDR_BA	MMIO	PE Packet Descriptor Ring Base Address Register	0x0 E010 0048	R/W	326
CRYP0_PE_PR_SCA	MMIO	PE Particle Descriptor Source Address Register	0x0 E010 0500	R/W	334
CRYP0_PE_PR_SCC	MMIO	PE Particle Descriptor Source Control Register	0x0 E010 0504	R/W	335
CRYP0_PE_PR_DTA	MMIO	PE Particle Descriptor Destination Address Register	0x0 E010 0580	R/W	336
CRYP0_PE_PR_DTC	MMIO	PE Particle Descriptor Destination Control Register	0x0 E010 0584	R/W	336
CRYP0_PE_PT_CFG	MMIO	PE Particle Ring Configuration Register	0x0 E010 0070	R/W	333
CRYP0_PE_PT_S	MMIO	PE Particle Ring Size Register	0x0 E010 006C	R/W	333
CRYP0_PE_RDR_BA	MMIO	PE Result Descriptor Ring Base Address Register	0x0 E010 004C	R/W	327
CRYP0_PE_RING_P	MMIO	PE Ring Poll Register	0x0 E010 0054	R/W	329
CRYP0_PE_RING_S	MMIO	PE Ring Size and Offset Register	0x0 E010 0050	R/W	328
CRYP0_PE_SA	MMIO	PE SA Address Register	0x0 E010 000C	R/W	320
CRYP0_PE_SCAT	MMIO	PE Scatter Particle Ring Base Address Register	0x0 E010 0068	R/W	332
CRYP0_PE_SOURCE	MMIO	PE Source Address Register	0x0 E010 0004	R/W	319
CRYP0_PKA_A_LEN	MMIO	PKA A Vector Length Register	0x0 E013 0810	R/W	365
CRYP0_PKA_B_LEN	MMIO	PKA B Vector Length Register	0x0 E013 0814	R/W	365
CRYP0_PKA_A_PTR	MMIO	PKA A Vector Address Register	0x0 E013 0800	R/W	365
CRYP0_PKA_B_PTR	MMIO	PKA B Vector Address Register	0x0 E013 0804	R/W	365
CRYP0_PKA_C_PTR	MMIO	PKA C Vector Address Register	0x0 E013 0808	R/W	365
CRYP0_PKA_COMP	MMIO	PKA Comparison Result Register	0x0 E013 0820	R	370
CRYP0_PKA_D_PTR	MMIO	PKA D Vector Address Register	0x0 E013 080C	R/W	365
CRYP0_PKA_DIV	MMIO	PKA Address of Quotient MSW Register	0x0 E013 0824	R	371
CRYP0_PKA_FUNC	MMIO	PKA Function Code Register	0x0 E013 081C	R/W	367
CRYP0_PKA_MOD	MMIO	PKA Address of Remainder MSW Register	0x0 E013 0828	R	372
CRYP0_PKA_SHIFT	MMIO	PKA Shift Register	0x0 E013 0818	R/W	366
CRYP0_PRNG_CTRL	MMIO	PRNG Control Register	0x0 E017 0004	R/W	384
CRYP0_PRNG_K0H	MMIO	PRNG Key 0 High Register	0x0 E017 0014	W	385
CRYP0_PRNG_K0L	MMIO	PRNG Key 0 Low Register	0x0 E017 0010	W	385
CRYP0_PRNG_K1H	MMIO	PRNG Key 1 High Register	0x0 E017 001C	W	385
CRYP0_PRNG_K1L	MMIO	PRNG Key 1 Low Register	0x0 E017 0018	W	385
CRYP0_PRNG_LFH	MMIO	PRNG LFSR High Register	0x0 E017 0034	R/W	386
CRYP0_PRNG_LFL	MMIO	PRNG LFSR Low Register	0x0 E017 0030	R/W	386
CRYP0_PRNG_RS0	MMIO	PRNG Result 0 (31:0) Register	0x0 E017 0020	R	386
CRYP0_PRNG_RS1	MMIO	PRNG Result 1 (63:32) Register	0x0 E017 0024	R	386
CRYP0_PRNG_RS2	MMIO	PRNG Result 2 (95:64) Register	0x0 E017 0028	R	386
CRYP0_PRNG_RS3	MMIO	PRNG Result 3 (127:96) Register	0x0 E017 002C	R	386

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
CRYP0_PRNG_SDH	MMIO	PRNG Seed Value High Register	0x0 E017 000C	W	385
CRYP0_PRNG_SDL	MMIO	PRNG Seed Value Low Register	0x0 E017 0008	W	385
CRYP0_PRNG_STAT	MMIO	PRNG Status Register	0x0 E017 0000	R	383
CRYP0_SA_ARC4IJ	MMIO	SA ARC4 i and j Pointer Register	0x0 E011 0670	R/W	352
CRYP0_SA_ARC4SB	MMIO	SA ARC4 State Address Pointer Register	0x0 E011 0674	R/W	353
CRYP0_SA_CMD_0	MMIO	See SA Command 0 Word Register	0x0 E010 0014	W	na
CRYP0_SA_CMD_1	MMIO	See SA Command 1 Word Register	0x0 E010 0018	W	na
CRYP0_SA_CMD_0	MMIO	SA Command 0 Word Register	0x0 E011 0600	W	338
CRYP0_SA_CMD_1	MMIO	SA Command 1 Word Register	0x0 E011 0604	W	342
CRYP0_SA_HASH_B	MMIO	SA Hash Byte Count Register	0x0 E011 06D0	R/W	354
CRYP0_SA_ICV_0	MMIO	SA ICV 0—HMAC result (outbound and inbound) Register	0x0 E011 06E8	R	355
CRYP0_SA_ICV_1	MMIO	SA ICV 1—HMAC result (outbound and inbound) Register	0x0 E011 06EC	R	355
CRYP0_SA_ICV_2	MMIO	SA ICV 2—HMAC result (outbound and inbound) Register	0x0 E011 06F0	R	355
CRYP0_SA_ICV_3	MMIO	SA ICV 3—HMAC result (outbound and inbound) Register	0x0 E011 06F4	R	355
CRYP0_SA_ICV_4	MMIO	SA ICV 4—HMAC result (outbound and inbound) Register	0x0 E011 06F8	R	355
CRYP0_SA_IH_0	MMIO	SA Inner Hash 0 (mirror of CRYP0_SA_IH_D0) Register	0x0 E011 06D4	See register	354
CRYP0_SA_IH_1	MMIO	SA Inner Hash 1 (mirror of CRYP0_SA_IH_D1) Register	0x0 E011 06D8	See register	354
CRYP0_SA_IH_2	MMIO	SA Inner Hash 2 (mirror of CRYP0_SA_IH_D2) Register	0x0 E011 06DC	See register	354
CRYP0_SA_IH_3	MMIO	SA Inner Hash 3 (mirror of CRYP0_SA_IH_D3) Register	0x0 E011 06E0	See register	354
CRYP0_SA_IH_4	MMIO	SA Inner Hash 4 (mirror of CRYP0_SA_IH_D4) Register	0x0 E011 06E4	R/W	354
CRYP0_SA_IH_D0	MMIO	SA Inner Hash Digest 0 Register	0x0 E011 0630	R/W	347
CRYP0_SA_IH_D1	MMIO	SA Inner Hash Digest 1 Register	0x0 E011 0634	R/W	347
CRYP0_SA_IH_D2	MMIO	SA Inner Hash Digest 2 Register	0x0 E011 0638	R/W	347
CRYP0_SA_IH_D3	MMIO	SA Inner Hash Digest 3 Register	0x0 E011 063C	R/W	347
CRYP0_SA_IH_D4	MMIO	SA Inner Hash Digest 4 Register	0x0 E011 0640	R/W	347
CRYP0_SA_IV_0	MMIO	SA Initialization Vector 0 Register	0x0 E011 06C0	R/W	353
CRYP0_SA_IV_1	MMIO	SA Initialization Vector 1 Register	0x0 E011 06C4	R/W	353
CRYP0_SA_IV_2	MMIO	SA Initialization Vector 2 Register	0x0 E011 06C8	R/W	353
CRYP0_SA_IV_3	MMIO	SA Initialization Vector 3 Register	0x0 E011 06CC	R/W	353
CRYP0_SA_KEY1_L	MMIO	SA Key 1 Low Register	0x0 E011 0610	W	346
CRYP0_SA_KEY1_H	MMIO	SA Key 1 High Register	0x0 E011 0614	W	346
CRYP0_SA_KEY2_L	MMIO	SA Key 2 Low Register	0x0 E011 0618	W	346
CRYP0_SA_KEY2_H	MMIO	SA Key 2 High Register	0x0 E011 061C	W	346
CRYP0_SA_KEY3_L	MMIO	SA Key 3 Low Register	0x0 E011 0620	W	346
CRYP0_SA_KEY3_H	MMIO	SA Key 3 High Register	0x0 E011 0624	W	346
CRYP0_SA_KEY4_L	MMIO	SA Key 4 Low Register	0x0 E011 0628	W	346
CRYP0_SA_KEY4_H	MMIO	SA Key 4 High Register	0x0 E011 062C	W	346
CRYP0_SA_NONCE	MMIO	SA Nonce Value Register	0x0 E011 0668	R/W	351

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
CRYP0_SA_OH_D0	MMIO	SA Outer Hash Digest 0 Register	0x0 E011 0644	W	348
CRYP0_SA_OH_D1	MMIO	SA Outer Hash Digest 1 Register	0x0 E011 0648	W	348
CRYP0_SA_OH_D2	MMIO	SA Outer Hash Digest 2 Register	0x0 E011 064C	W	348
CRYP0_SA_OH_D3	MMIO	SA Outer Hash Digest 3 Register	0x0 E011 0650	W	348
CRYP0_SA_OH_D4	MMIO	SA Outer Hash Digest 4 Register	0x0 E011 0654	W	348
CRYP0_SA_PNTR	MMIO	SA Pointer Register	0x0 E011 066C	R/W	352
CRYP0_SA_SEQ	MMIO	SA IPsec Sequence Number Register	0x0 E011 065C	R/W	350
CRYP0_SA_SEQMKH	MMIO	SA IPsec Sequence Number Mask High Register	0x0 E011 0664	R/W	351
CRYP0_SA_SEQMKL	MMIO	SA IPsec Sequence Number Mask Low Register	0x0 E011 0660	R/W	351
CRYP0_SA_SPI	MMIO	SA IPsec SPI Register	0x0 E011 0658	R/W	349
CRYP0_TRNG_ALRM	MMIO	TRNG Alarm Count Register	0x0 E012 0124	R	360
CRYP0_TRNG_CFG	MMIO	TRNG Configuration Register	0x0 E012 0128	R/W	361
CRYP0_TRNG_CNTR	MMIO	TRNG Counter Register	0x0 E012 0120	R/W	360
CRYP0_TRNG_CTRL	MMIO	TRNG Test Control Register	0x0 E012 0108	R/W	357
CRYP0_TRNG_DATA	MMIO	TRNG Output Register	0x0 E012 0100	R/W	356
CRYP0_TRNG_ENTA	MMIO	TRNG Test Entropy A Register	0x0 E012 010C	R/W	358
CRYP0_TRNG_ENTB	MMIO	TRNG Test Entropy B Register	0x0 E012 0110	R/W	359
CRYP0_TRNG_IV_H	MMIO	TRNG Initialization Vector High Register	0x0 E012 0150	W	364
CRYP0_TRNG_IV_L	MMIO	TRNG Initialization Vector Low Register	0x0 E012 014C	W	364
CRYP0_TRNG_K0_H	MMIO	TRNG Triple DES Key 0 High Register	0x0 E012 0140	W	363
CRYP0_TRNG_K0_L	MMIO	TRNG Triple DES Key 0 Low Register	0x0 E012 013C	W	363
CRYP0_TRNG_K1_H	MMIO	TRNG Triple DES Key 1 High Register	0x0 E012 0148	W	364
CRYP0_TRNG_K1_L	MMIO	TRNG Triple DES Key 1 Low Register	0x0 E012 0144	W	364
CRYP0_TRNG_LF0H	MMIO	TRNG Test Read of LFSR0 High Register	0x0 E012 0130	R	362
CRYP0_TRNG_LF0L	MMIO	TRNG Test Read of LFSR 0 Low Register	0x0 E012 012C	R	362
CRYP0_TRNG_LF1H	MMIO	TRNG Test Read of LFSR1 High Register	0x0 E012 0138	R	363
CRYP0_TRNG_LF1L	MMIO	TRNG Test Read of LFSR1 Low Register	0x0 E012 0134	R	363
CRYP0_TRNG_STAT	MMIO	TRNG Status Register	0x0 E012 0104	R	356
CRYP0_TRNG_X0	MMIO	TRNG Test Seed 0 Register	0x0 E012 0114	R/W	359
CRYP0_TRNG_X1	MMIO	TRNG Test Seed 1 Register	0x0 E012 0118	R/W	359
CRYP0_TRNG_X2	MMIO	TRNG Test Seed 2 Register	0x0 E012 011C	R/W	359
DDR0_CFGADDR	DCR	DDR-SDRAM Address Register	0x0010	W	410
DDR0_CFGDATA	DCR	DDR-SDRAM Data Register	0x0011	R/W	410
DDR0_00	DCR	DDR-SDRAM Register	offset 0x00	See register	412
DDR0_01	DCR	DDR-SDRAM Register	offset 0x01	See register	412
DDR0_02	DCR	DDR-SDRAM Register	offset 0x02	See register	412
DDR0_03	DCR	DDR-SDRAM Register	offset 0x03	R/W	412
DDR0_04	DCR	DDR-SDRAM Register	offset 0x04	R/W	412

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
DDR0_05	DCR	DDR-SDRAM Register	offset 0x05	R/W	412
DDR0_06	DCR	DDR-SDRAM Register	offset 0x06	R/W	412
DDR0_07	DCR	DDR-SDRAM Register	offset 0x07	See register	412
DDR0_08	DCR	DDR-SDRAM Register	offset 0x08	R/W	412
DDR0_09	DCR	DDR-SDRAM Register	offset 0x09	R/W	412
DDR0_10	DCR	DDR-SDRAM Register	offset 0x0A	See register	412
DDR0_11	DCR	DDR-SDRAM Register	offset 0x0B	R/W	412
DDR0_12	DCR	DDR-SDRAM Register	offset 0x0C	R/W	412
DDR0_14	DCR	DDR-SDRAM Register	offset 0x0E	R/W	412
DDR0_17	DCR	DDR-SDRAM Register	offset 0x11	R/W	412
DDR0_18	DCR	DDR-SDRAM Register	offset 0x12	R/W	412
DDR0_19	DCR	DDR-SDRAM Register	offset 0x13	R/W	412
DDR0_20	DCR	DDR-SDRAM Register	offset 0x14	R/W	412
DDR0_21	DCR	DDR-SDRAM Register	offset 0x15	R/W	412
DDR0_22	DCR	DDR-SDRAM Register	offset 0x16	R/W	412
DDR0_23	DCR	DDR-SDRAM Register	offset 0x17	See register	412
DDR0_24	DCR	DDR-SDRAM Register	offset 0x18	R/W	412
DDR0_25	DCR	DDR-SDRAM Register	offset 0x19	R/W	412
DDR0_26	DCR	DDR-SDRAM Register	offset 0x1A	R/W	412
DDR0_27	DCR	DDR-SDRAM Register	offset 0x1B	R/W	412
DDR0_28	DCR	DDR-SDRAM Register	offset 0x1C	R/W	412
DDR0_31	DCR	DDR-SDRAM Register	offset 0x1F	R/W	412
DDR0_32	DCR	DDR-SDRAM Register	offset 0x20	R	412
DDR0_33	DCR	DDR-SDRAM Register	offset 0x21	R	412
DDR0_34	DCR	DDR-SDRAM Register	offset 0x22	R	412
DDR0_35	DCR	DDR-SDRAM Register	offset 0x23	R	412
DDR0_36	DCR	DDR-SDRAM Register	offset 0x24	R	412
DDR0_37	DCR	DDR-SDRAM Register	offset 0x25	R	412
DDR0_38	DCR	DDR-SDRAM Register	offset 0x26	R	412
DDR0_39	DCR	DDR-SDRAM Register	offset 0x27	R	412
DDR0_40	DCR	DDR-SDRAM Register	offset 0x28	R	412
DDR0_41	DCR	DDR-SDRAM Register	offset 0x29	R	412
DDR0_42	DCR	DDR-SDRAM Register	offset 0x2A	R/W	412
DDR0_43	DCR	DDR-SDRAM Register	offset 0x2B	R/W	412
DDR0_44	DCR	DDR-SDRAM Register	offset 0x2C	R/W	412
DMA2P30_ADR	DCR	DMA to PLB 3 Address Decode Register	0x0124	R/W	636
DMA2P30_CR0	DCR	DMA to PLB 3 Channel Control Register 0	0x0100	R/W	632
DMA2P30_CR1	DCR	DMA to PLB 3 Channel Control Register 1	0x0108	R/W	632

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
DMA2P30_CR2	DCR	DMA to PLB 3 Channel Control Register 2	0x0110	R/W	632
DMA2P30_CR3	DCR	DMA to PLB 3 Channel Control Register 3	0x0118	R/W	632
DMA2P30_CT0	DCR	DMA to PLB 3 Count Register 0	0x0101	R/W	634
DMA2P30_CT1	DCR	DMA to PLB 3 Count Register 1	0x0109	R/W	634
DMA2P30_CT2	DCR	DMA to PLB 3 Count Register 2	0x0111	R/W	634
DMA2P30_CT3	DCR	DMA to PLB 3 Count Register 3	0x0119	R/W	634
DMA2P30_DA0	DCR	DMA to PLB 3 Destination Address Register 0	0x0102	R/W	634
DMA2P30_DA1	DCR	DMA to PLB 3 Destination Address Register 1	0x010A	R/W	634
DMA2P30_DA2	DCR	DMA to PLB 3 Destination Address Register 2	0x0112	R/W	634
DMA2P30_DA3	DCR	DMA to PLB 3 Destination Address Register 3	0x011A	R/W	634
DMA2P30_POL	DCR	DMA to PLB 3 Polarity Configuration Register	0x0126	R/W	630
DMA2P30_SA0	DCR	DMA to PLB 3 Source Address Register 0	0x0103	R/W	634
DMA2P30_SA1	DCR	DMA to PLB 3 Source Address Register 1	0x010B	R/W	634
DMA2P30_SA2	DCR	DMA to PLB 3 Source Address Register 2	0x0113	R/W	634
DMA2P30_SA3	DCR	DMA to PLB 3 Source Address Register 3	0x011B	R/W	634
DMA2P30_SC0	DCR	DMA to PLB 3 Subchannel ID Register 0	0x0107	R/W	636
DMA2P30_SC1	DCR	DMA to PLB 3 Subchannel ID Register 1	0x010F	R/W	636
DMA2P30_SC2	DCR	DMA to PLB 3 Subchannel ID Register 2	0x0117	R/W	636
DMA2P30_SC3	DCR	DMA to PLB 3 Subchannel ID Register 3	0x011F	R/W	636
DMA2P30_SG0	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 0	0x0104	R/W	635
DMA2P30_SG1	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 1	0x010C	R/W	635
DMA2P30_SG2	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 2	0x0114	R/W	635
DMA2P30_SG3	DCR	DMA to PLB 3 Scatter/Gather Descriptor Address Register 3	0x011C	R/W	635
DMA2P30_SGC	DCR	DMA to PLB 3 Scatter/Gather Command Register	0x0123	R/W	635
DMA2P30_SLP	DCR	DMA to PLB 3 Sleep Mode Register	0x0125	R/W	631
DMA2P30_SR	DCR	DMA to PLB 3 Status Register	0x0120	R/W	631
DMA2P40_CR0	DCR	DMA to PLB 4 Channel Control Register 0	0x0300	R/W	613
DMA2P40_CR1	DCR	DMA to PLB 4 Channel Control Register 1	0x0308	R/W	613
DMA2P40_CR2	DCR	DMA to PLB 4 Channel Control Register 2	0x0310	R/W	613
DMA2P40_CR3	DCR	DMA to PLB 4 Channel Control Register 3	0x0318	R/W	613
DMA2P40_CT0	DCR	DMA to PLB 4 Count and Control Register 0	0x0301	R/W	614
DMA2P40_CT1	DCR	DMA to PLB 4 Count and Control Register 1	0x0309	R/W	614
DMA2P40_CT2	DCR	DMA to PLB 4 Count and Control Register 2	0x0311	R/W	614
DMA2P40_CT3	DCR	DMA to PLB 4 Count and Control Register 3	0x0319	R/W	614
DMA2P40_DAH0	DCR	DMA to PLB 4 Destination Address High Register 0	0x0304	R/W	616
DMA2P40_DAH1	DCR	DMA to PLB 4 Destination Address High Register 1	0x030C	R/W	616
DMA2P40_DAH2	DCR	DMA to PLB 4 Destination Address High Register 2	0x0314	R/W	616
DMA2P40_DAH3	DCR	DMA to PLB 4 Destination Address High Register 3	0x031C	R/W	616

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
DMA2P40_DAL0	DCR	DMA to PLB 4 Destination Address Low Register 0	0x0305	R/W	616
DMA2P40_DAL1	DCR	DMA to PLB 4 Destination Address Low Register 1	0x030D	R/W	616
DMA2P40_DAL2	DCR	DMA to PLB 4 Destination Address Low Register 2	0x0315	R/W	616
DMA2P40_DAL3	DCR	DMA to PLB 4 Destination Address Low Register 3	0x031D	R/W	616
DMA2P40_POL	DCR	DMA to PLB 4 Polarity Configuration Register	0x0326	R/W	619
DMA2P40_SAH0	DCR	DMA to PLB 4 Source Address High Register 0	0x0302	R/W	615
DMA2P40_SAH1	DCR	DMA to PLB 4 Source Address High Register 1	0x030A	R/W	615
DMA2P40_SAH2	DCR	DMA to PLB 4 Source Address High Register 2	0x0312	R/W	615
DMA2P40_SAH3	DCR	DMA to PLB 4 Source Address High Register 3	0x031A	R/W	615
DMA2P40_SAL0	DCR	DMA to PLB 4 Source Address Low Register 0	0x0303	R/W	615
DMA2P40_SAL1	DCR	DMA to PLB 4 Source Address Low Register 1	0x030B	R/W	615
DMA2P40_SAL2	DCR	DMA to PLB 4 Source Address Low Register 2	0x0313	R/W	615
DMA2P40_SAL3	DCR	DMA to PLB 4 Source Address Low Register 3	0x031B	R/W	615
DMA2P40_SGC	DCR	DMA to PLB 4 Scatter/Gather Command Register	0x0323	R/W	618
DMA2P40_SGH0	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 0	0x0306	R/W	616
DMA2P40_SGH1	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 1	0x030E	R/W	616
DMA2P40_SGH2	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 2	0x0316	R/W	616
DMA2P40_SGH3	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address High 3	0x031E	R/W	616
DMA2P40_SGL0	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 0	0x0307	R/W	616
DMA2P40_SGL1	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 1	0x030F	R/W	616
DMA2P40_SGL2	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 2	0x0317	R/W	616
DMA2P40_SGL3	DCR	DMA to PLB 4 Scatter/Gather Descriptor Address Low 3	0x031F	R/W	616
DMA2P40_SLP	DCR	DMA to PLB 4 Sleep Mode Register	0x0325	R/W	618
DMA2P40_SR	DCR	DMA to PLB 4 Status Register	0x0320	R/W	617
EBC0_B0AP	DCR	Peripheral Bank 0 Access Parameters	offset 0x0010	R/W	578
EBC0_B0CR	DCR	Peripheral Bank 0 Configuration Register	offset 0x0000	R/W	577
EBC0_B1AP	DCR	Peripheral Bank 1 Access Parameters	offset 0x0011	R/W	578
EBC0_B1CR	DCR	Peripheral Bank 1 Configuration Register	offset 0x0001	R/W	577
EBC0_B2AP	DCR	Peripheral Bank 2 Access Parameters	offset 0x0012	R/W	578
EBC0_B2CR	DCR	Peripheral Bank 2 Configuration Register	offset 0x0002	R/W	577
EBC0_B3AP	DCR	Peripheral Bank 3 Access Parameters	offset 0x0013	R/W	578
EBC0_B3CR	DCR	Peripheral Bank 3 Configuration Register	offset 0x0003	R/W	577
EBC0_B4AP	DCR	Peripheral Bank 4 Access Parameters	offset 0x0014	R/W	578
EBC0_B4CR	DCR	Peripheral Bank 4 Configuration Register	offset 0x0004	R/W	577
EBC0_B5AP	DCR	Peripheral Bank 5 Access Parameters	offset 0x0015	R/W	578
EBC0_B5CR	DCR	Peripheral Bank 5 Configuration Register	offset 0x0005	R/W	577
EBC0_BEAR	DCR	Peripheral Bus Error Address Register	offset 0x0020	R/W	580
EBC0_BESR0	DCR	Peripheral Bus Error Status Register 0	offset 0x0021	R/W	580

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
EBC0_CFG	DCR	External Peripheral Control Register	offset 0x0023	R/W	576
EBC0_CFGADDR	DCR	Peripheral Bank Address Register	0x0012	R/W	574
EBC0_CFGDATA	DCR	Peripheral Bank Data Register	0x0013	R/W	574
EHCI0_HCCAPBASE	MMIO	Version/Capability Length	0x0 E000 0300	R	818
EHCI0_HCSPARAMS	MMIO	Structural Parameters	0x0 E000 0304	R	818
EHCI0_HCCPARAMS	MMIO	Capability Parameters	0x0 E000 0308	R	818
EHCI0_USBCMD	MMIO	USB Command	0x0 E000 0310	R/W	818
EHCI0_USBSTS	MMIO	USB Status	0x0 E000 0314	R/W	818
EHCI0_USBINTR	MMIO	USB Interrupt Enable	0x0 E000 0318	R/W	818
EHCI0_FRINDEX	MMIO	EHCI Frame Index	0x0 E000 031C	R/W	818
EHCI0_CRTLSEG	MMIO	Control Data Structure Segment	0x0 E000 0320	R/W	818
EHCI0_PRDLISTB	MMIO	Periodic Frame List Base Address	0x0 E000 0324	R/W	818
EHCI0_ASYNCLSTA	MMIO	Current Asynchronous List Address	0x0 E000 0328	R/W	818
EHCI0_CFGFLAG	MMIO	Configure Flag	0x0 E000 0350	R/W	818
EHCI0_PORTSC	MMIO	Port Status and Control	0x0 E000 0354	R/W	818
EHCI0_INSNREG0	MMIO	Programmable Microframe Base Value	0x0 E000 0390	R/W	818
EHCI0_INSNREG1	MMIO	Programmable Packet Buffer Threshold	0x0 E000 0394	R/W	818
EHCI0_INSNREG2	MMIO	Programmable Packet Buffer Depth	0x0 E000 0398	R/W	818
EHCI0_INSNREG3	MMIO	Break Memory Transfer	0x0 E000 039C	R/W	818
EHCI0_INSNREG4	MMIO	Debug Only	0x0 E000 03A0	R/W	818
EMAC0_GAHT1	MMIO	EMAC 0 Group Address Hash Table 1	0x1 EF60 0E40	R/W	734
EMAC0_GAHT2	MMIO	EMAC 0 Group Address Hash Table 2	0x1 EF60 0E44	R/W	734
EMAC0_GAHT3	MMIO	EMAC 0 Group Address Hash Table 3	0x1 EF60 0E48	R/W	734
EMAC0_GAHT4	MMIO	EMAC 0 Group Address Hash Table 4	0x1 EF60 0E4C	R/W	734
EMAC0_IAHR	MMIO	EMAC 0 Individual Address High	0x1 EF60 0E1C	R/W	732
EMAC0_IAHT1	MMIO	EMAC 0 Individual Address Hash Table 1	0x1 EF60 0E30	R/W	734
EMAC0_IAHT2	MMIO	EMAC 0 Individual Address Hash Table 2	0x1 EF60 0E34	R/W	734
EMAC0_IAHT3	MMIO	EMAC 0 Individual Address Hash Table 3	0x1 EF60 0E38	R/W	734
EMAC0_IAHT4	MMIO	EMAC 0 Individual Address Hash Table 4	0x1 EF60 0E3C	R/W	734
EMAC0_IALR	MMIO	EMAC 0 Individual Address Low	0x1 EF60 0E20	R/W	732
EMAC0_IPGVR	MMIO	EMAC 0 Inter-Packet Gap Value Register	0x1 EF60 0E58	R/W	735
EMAC0_ISER	MMIO	EMAC 0 Interrupt Status Enable Register	0x1 EF60 0E18	R/W	730
EMAC0_ISR	MMIO	EMAC 0 Interrupt Status Register	0x1 EF60 0E14	R/W	734
EMAC0_LSAH	MMIO	EMAC 0 Last Source Address Low	0x1 EF60 0E50	R	734
EMAC0_LSAL	MMIO	EMAC 0 Last Source Address High	0x1 EF60 0E54	R	734
EMAC0_MR0	MMIO	EMAC 0 Mode Register 0	0x1 EF60 0E00	R/W	723
EMAC0_MR1	MMIO	EMAC 0 Mode Register 1	0x1 EF60 0E04	R/W	723
EMAC0_OCRX	MMIO	EMAC 0 Number of Octets Received	0x1 EF60 0E6C	R	738

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
EMAC0_OCTX	MMIO	EMAC 0 Number of Octets Transmitted	0x1 EF60 0E68	R	738
EMAC0_PTR	MMIO	EMAC 0 Pause Timer Register	0x1 EF60 0E2C	R/W	733
EMAC0_RMR	MMIO	EMAC 0 Receive Mode Register	0x1 EF60 0E10	R/W	726
EMAC0_RWMR	MMIO	EMAC 0 Receive Low/High Water Mark Register	0x1 EF60 0E64	R/W	737
EMAC0_STACR	MMIO	EMAC 0 STA Control Register	0x1 EF60 0E5C	R/W	735
EMAC0_TMR0	MMIO	EMAC 0 Transmit Mode Register 0	0x1 EF60 0E08	R/W	725
EMAC0_TMR1	MMIO	EMAC 0 Transmit Mode Register 1	0x1 EF60 0E0C	R/W	726
EMAC0_TRTR	MMIO	EMAC 0 Transmit Request Threshold Register	0x1 EF60 0E60	R/W	736
EMAC0_VTCI	MMIO	EMAC 0 VLAN TCI Register	0x1 EF60 0E28	R/W	733
EMAC0_VTPID	MMIO	EMAC 0 VLAN TPID Register	0x1 EF60 0E24	R/W	733
EMAC1_GAHT1	MMIO	EMAC 1 Group Address Hash Table 1	0x1 EF60 0F40	R/W	734
EMAC1_GAHT2	MMIO	EMAC 1 Group Address Hash Table 2	0x1 EF60 0F44	R/W	734
EMAC1_GAHT3	MMIO	EMAC 1 Group Address Hash Table 3	0x1 EF60 0F48	R/W	734
EMAC1_GAHT4	MMIO	EMAC 1 Group Address Hash Table 4	0x1 EF60 0F4C	R/W	734
EMAC1_IAHR	MMIO	EMAC 1 Individual Address High	0x1 EF60 0F1C	R/W	732
EMAC1_IAHT1	MMIO	EMAC 1 Individual Address Hash Table 1	0x1 EF60 0F30	R/W	734
EMAC1_IAHT2	MMIO	EMAC 1 Individual Address Hash Table 2	0x1 EF60 0F34	R/W	734
EMAC1_IAHT3	MMIO	EMAC 1 Individual Address Hash Table 3	0x1 EF60 0F38	R/W	734
EMAC1_IAHT4	MMIO	EMAC 1 Individual Address Hash Table 4	0x1 EF60 0F3C	R/W	734
EMAC1_IALR	MMIO	EMAC 1 Individual Address Low	0x1 EF60 0F20	R/W	732
EMAC1_IPGVR	MMIO	EMAC 1 Inter-Packet Gap Value Register	0x1 EF60 0F58	R/W	735
EMAC1_ISER	MMIO	EMAC 1 Interrupt Status Enable Register	0x1 EF60 0F18	R/W	730
EMAC1_ISR	MMIO	EMAC 1 Interrupt Status Register	0x1 EF60 0F14	R/W	734
EMAC1_LSAH	MMIO	EMAC 1 Last Source Address Low	0x1 EF60 0F50	R	734
EMAC1_LSAL	MMIO	EMAC 1 Last Source Address High	0x1 EF60 0F54	R	734
EMAC1_MR0	MMIO	EMAC 1 Mode Register 0	0x1 EF60 0F00	R/W	723
EMAC1_MR1	MMIO	EMAC 1 Mode Register 1	0x1 EF60 0F04	R/W	723
EMAC1_OCRX	MMIO	EMAC 1 Number of Octets Received	0x1 EF60 0F6C	R	738
EMAC1_OCTX	MMIO	EMAC 1 Number of Octets Transmitted	0x1 EF60 0F68	R	738
EMAC1_PTR	MMIO	EMAC 1 Pause Timer Register	0x1 EF60 0F2C	R/W	733
EMAC1_RMR	MMIO	EMAC 1 Receive Mode Register	0x1 EF60 0F10	R/W	726
EMAC1_RWMR	MMIO	EMAC 1 Receive Low/High Water Mark Register	0x1 EF60 0F64	R/W	737
EMAC1_STACR	MMIO	EMAC 1 STA Control Register	0x1 EF60 0F5C	R/W	735
EMAC1_TMR0	MMIO	EMAC 1 Transmit Mode Register 0	0x1 EF60 0F08	R/W	725
EMAC1_TMR1	MMIO	EMAC 1 Transmit Mode Register 1	0x1 EF60 0F0C	R/W	726
EMAC1_TRTR	MMIO	EMAC 1 Transmit Request Threshold Register	0x1 EF60 0F60	R/W	736
EMAC1_VTCI	MMIO	EMAC 1 VLAN TCI Register	0x1 EF60 0F28	R/W	733
EMAC1_VTPID	MMIO	EMAC 1 VLAN TPID Register	0x1 EF60 0F24	R/W	733

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
FPR0	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR1	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR2	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR3	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR4	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR5	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR6	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR7	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR8	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR9	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR10	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR11	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR12	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR13	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR14	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR15	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR16	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR17	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR18	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR19	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR20	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR21	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR22	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR23	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR24	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR25	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR26	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR27	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR28	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR29	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR30	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPR31	FPR	Floating Point Register *(PPC440EPx only)*	na	na	142
FPSCR	FPR	Floating Point Status and Control Register	na	na	142
GPIO0_IR	MMIO	GPIO0 Input Register	0x1 EF60 0B1C	R	801
GPIO0_ISR1H	MMIO	GPIO0 Input Select Register 1 High	0x1 EF60 0B34	R/W	801
GPIO0_ISR1L	MMIO	GPIO0 Input Select Register 1 Low	0x1 EF60 0B30	R/W	801
GPIO0_ISR2H	MMIO	GPIO0 Input Select Register 2 High	0x1 EF60 0B3C	R/W	801
GPIO0_ISR2L	MMIO	GPIO0 Input Select Register 2 Low	0x1 EF60 0B38	R/W	801

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
GPIO0_ISR3H	MMIO	GPIO0 Input Select Register 3 High	0x1 EF60 0B44	R/W	801
GPIO0_ISR3L	MMIO	GPIO0 Input Select Register 3 Low	0x1 EF60 0B40	R/W	801
GPIO0_ODR	MMIO	GPIO0 Open Drain Register	0x1 EF60 0B18	R/W	800
GPIO0_OR	MMIO	GPIO0 Output	0x1 EF60 0B00	R/W	799
GPIO0_OSRH	MMIO	GPIO0 Output Select Register High	0x1 EF60 0B0C	R/W	799
GPIO0_OSRL	MMIO	GPIO0 Output Select Register Low	0x1 EF60 0B08	R/W	799
GPIO0_RR1	MMIO	GPIO0 Receive Register 1	0x1 EF60 0B20	R/W	802
GPIO0_RR2	MMIO	GPIO0 Receive Register 2	0x1 EF60 0B24	R/W	802
GPIO0_RR3	MMIO	GPIO0 Receive Register 3	0x1 EF60 0B28	R/W	802
GPIO0_TCR	MMIO	GPIO0 Three-State Control Register	0x1 EF60 0B04	R/W	799
GPIO0_TSRH	MMIO	GPIO0 Three-State Select Register High	0x1 EF60 0B14	R/W	800
GPIO0_TSRL	MMIO	GPIO0 Three-State Select Register Low	0x1 EF60 0B10	R/W	800
GPIO1_IR	MMIO	GPIO1 Input Register	0x1 EF60 0C1C	R	801
GPIO1_ISR1H	MMIO	GPIO1 Input Select Register 1 High	0x1 EF60 0C34	R/W	801
GPIO1_ISR1L	MMIO	GPIO1 Input Select Register 1 Low	0x1 EF60 0C30	R/W	801
GPIO1_ISR2H	MMIO	GPIO1 Input Select Register 2 High	0x1 EF60 0C3C	R/W	801
GPIO1_ISR2L	MMIO	GPIO1 Input Select Register 2 Low	0x1 EF60 0C38	R/W	801
GPIO1_ISR3H	MMIO	GPIO1 Input Select Register 3 High	0x1 EF60 0C44	R/W	801
GPIO1_ISR3L	MMIO	GPIO1 Input Select Register 3 Low	0x1 EF60 0C40	R/W	801
GPIO1_ODR	MMIO	GPIO1 Open Drain Register	0x1 EF60 0C18	R/W	800
GPIO1_OR	MMIO	GPIO1 Output Register	0x1 EF60 0C00	R/W	799
GPIO1_OSRH	MMIO	GPIO1 Output Select Register High	0x1 EF60 0C0C	R/W	799
GPIO1_OSRL	MMIO	GPIO1 Output Select Register Low	0x1 EF60 0C08	R/W	799
GPIO1_RR1	MMIO	GPIO1 Receive Register 1	0x1 EF60 0C20	R/W	802
GPIO1_RR2	MMIO	GPIO1 Receive Register 2	0x1 EF60 0C24	R/W	802
GPIO1_RR3	MMIO	GPIO1 Receive Register 3	0x1 EF60 0C28	R/W	802
GPIO1_TCR	MMIO	GPIO1 Three-State Control Register	0x1 EF60 0C04	R/W	799
GPIO1_TSRH	MMIO	GPIO1 Three-State Select Register High	0x1 EF60 0C14	R/W	800
GPIO1_TSRL	MMIO	GPIO1 Three-State Select Register Low	0x1 EF60 0C10	R/W	800
GPT0_COMP0	MMIO	GPT Compare Timer 0	0x1 EF60 0080	R/W	270
GPT0_COMP1	MMIO	GPT Compare Timer 1	0x1 EF60 0084	R/W	270
GPT0_COMP2	MMIO	GPT Compare Timer 2	0x1 EF60 0088	R/W	270
GPT0_COMP3	MMIO	GPT Compare Timer 3	0x1 EF60 008C	R/W	270
GPT0_COMP4	MMIO	GPT Compare Timer 4	0x1 EF60 0090	R/W	270
GPT0_COMP5	MMIO	GPT Compare Timer 5	0x1 EF60 0094	R/W	270
GPT0_COMP6	MMIO	GPT Compare Timer 6	0x1 EF60 0098	R/W	270
GPT0_DCIS	MMIO	Down Count Timer Interrupt Status	0x1 EF60 011C	R/W	271
GPT0_DCT0	MMIO	Down Count Timer	0x1 EF60 0110	R/W	270

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
GPT0_IE	MMIO	GPT Interrupt Enable	0x1 EF60 0024	R/W	269
GPT0_IM	MMIO	GPT Interrupt Mask	0x1 EF60 0018	R/W	268
GPT0_ISC	MMIO	GPT Interrupt Status (Clear bits if write 1)	0x1 EF60 0020	R/W	268
GPT0_ISS	MMIO	GPT Interrupt Status (Set bits if write 1)	0x1 EF60 001C	R/W	268
GPT0_MASK0	MMIO	GPT Compare Mask 0	0x1 EF60 00C0	R/W	270
GPT0_MASK1	MMIO	GPT Compare Mask 1	0x1 EF60 00C4	R/W	270
GPT0_MASK2	MMIO	GPT Compare Mask 2	0x1 EF60 00C8	R/W	270
GPT0_MASK3	MMIO	GPT Compare Mask 3	0x1 EF60 00CC	R/W	270
GPT0_MASK4	MMIO	GPT Compare Mask 4	0x1 EF60 00D0	R/W	270
GPT0_MASK5	MMIO	GPT Compare Mask 5	0x1 EF60 00D4	R/W	270
GPT0_MASK6	MMIO	GPT Compare Mask 6	0x1 EF60 00D8	R/W	270
GPT0_TBC	MMIO	GPT Time Base Counter	0x1 EF60 0000	R/W	267
IIC0_CLKDIV	MMIO	IIC 0 Clock Divide	0x1 EF60 070C	R/W	786
IIC0_CNTL	MMIO	IIC 0 Control	0x1 EF60 0706	R/W	780
IIC0_DIRECTCNTL	MMIO	IIC 0 Direct Control	0x1 EF60 0710	R/W	790
IIC0_EXTSTS	MMIO	IIC 0 Extended Status	0x1 EF60 0709	R/W	784
IIC0_HMADR	MMIO	IIC 0 High Master Address	0x1 EF60 0705	R/W	780
IIC0_HSADR	MMIO	IIC 0 High Slave Address	0x1 EF60 070B	R/W	786
IIC0_INTR	MMIO	IIC 0 Interrupt	0x1 EF60 0711	R	791
IIC0_INTRMSK	MMIO	IIC 0 Interrupt Mask	0x1 EF60 070D	R/W	787
IIC0_LMADR	MMIO	IIC 0 Low Master Address	0x1 EF60 0704	R/W	779
IIC0_LSADR	MMIO	IIC 0 Low Slave Address	0x1 EF60 070A	R/W	785
IIC0_MDBUF	MMIO	IIC 0 Master Data Buffer	0x1 EF60 0700	R/W	778
IIC0_MDCNTL	MMIO	IIC 0 Mode Control	0x1 EF60 0707	R/W	781
IIC0_SDBUF	MMIO	IIC 0 Slave Data Buffer	0x1 EF60 0702	R/W	778
IIC0_STS	MMIO	IIC 0 Status	0x1 EF60 0708	R/W	782
IIC0_XFRCNT	MMIO	IIC 0 Transfer Count	0x1 EF60 070E	R/W	788
IIC0_XTCNTLSS	MMIO	IIC 0 Extended Control and Slave Status	0x1 EF60 070F	R/W	789
IIC1_CLKDIV	MMIO	IIC 1 Clock Divide	0x1 EF60 080C	R/W	786
IIC1_CNTL	MMIO	IIC 1 Control	0x1 EF60 0806	R/W	780
IIC1_DIRECTCNTL	MMIO	IIC 1 Direct Control	0x1 EF60 0810	R/W	790
IIC1_EXTSTS	MMIO	IIC 1 Extended Status	0x1 EF60 0809	R/W	784
IIC1_HMADR	MMIO	IIC 1 High Master Address	0x1 EF60 0805	R/W	780
IIC1_HSADR	MMIO	IIC 1 High Slave Address	0x1 EF60 080B	R/W	786
IIC1_INTR	MMIO	IIC 1 Interrupt	0x1 EF60 0811	R	791
IIC1_INTRMSK	MMIO	IIC 1 Interrupt Mask	0x1 EF60 080D	R/W	787
IIC1_LMADR	MMIO	IIC 1 Low Master Address	0x1 EF60 0804	R/W	779
IIC1_LSADR	MMIO	IIC 1 Low Slave Address	0x1 EF60 080A	R/W	785

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
IIC1_MDBUF	MMIO	IIC 1 Master Data Buffer	0x1 EF60 0800	R/W	778
IIC1_MDCNTL	MMIO	IIC 1 Mode Control	0x1 EF60 0807	R/W	781
IIC1_SDBUF	MMIO	IIC 1 Slave Data Buffer	0x1 EF60 0802	R/W	778
IIC1_STS	MMIO	IIC 1 Status	0x1 EF60 0808	R/W	782
IIC1_XFRCNT	MMIO	IIC 1 Transfer Count	0x1 EF60 080E	R/W	788
IIC1_XTCNTLSS	MMIO	IIC 1 Extended Control and Slave Status	0x1 EF60 080F	R/W	789
KASU0_CONFIG	MMIO	KASUMI Configuration	0x0 E018 0024	W	398
KASU0_COUNT	MMIO	KASUMI Count	0x0 E018 0020	W	398
KASU0_CTRL	MMIO	KASUMI Control	0x0 E018 0008	W	395
KASU0_DATAIN0	MMIO	KASUMI Data Input 0	0x0 E018 0000	R	394
KASU0_DATAIN1	MMIO	KASUMI Data Input 1	0x0 E018 0004	W	394
KASU0_DATAOUT0	MMIO	KASUMI Data Output 0	0x0 E018 0000	R	394
KASU0_DATAOUT1	MMIO	KASUMI Data Output 1	0x0 E018 0004	W	394
KASU0_FRESH	MMIO	KASUMI Fresh	0x0 E018 0028	W	399
KASU0_KEY0	MMIO	KASUMI Key 0	0x0 E018 0010	W	397
KASU0_KEY1	MMIO	KASUMI Key 1	0x0 E018 0014	W	397
KASU0_KEY2	MMIO	KASUMI Key 2	0x0 E018 0018	W	397
KASU0_KEY3	MMIO	KASUMI Key 3	0x0 E018 001C	W	397
KASU0_STAT	MMIO	KASUMI Status	0x0 E018 0008	R	395
MAL0_CFG	DCR	MAL Configuration Register	0x0180	R/W	669
MAL0_ESR	DCR	MAL Error Status Register	0x0181	R/Clear	672
MAL0_IER	DCR	MAL Interrupt Enable Register	0x0182	R/W	674
MAL0_RCBS0	DCR	MAL RX Channel 0 Buffer Size Register	0x01E0	R/W	677
MAL0_RCBS1	DCR	MAL RX Channel 1 Buffer Size Register	0x01E1	R/W	677
MAL0_RXBADDR	DCR	MAL Rx Descriptor Base Address Register	0x0195	R/W	676
MAL0_RXCARR	DCR	MAL Rx Channel Active Register (Reset)	0x0191	R/W	670
MAL0_RXCASR	DCR	MAL Rx Channel Active Register (Set)	0x0190	R/W	670
MAL0_RXCTP0R	DCR	MAL Rx Channel 0 Table Pointer Register	0x01C0	R/W	676
MAL0_RXCTP1R	DCR	MAL Rx Channel 1 Table Pointer Register	0x01C1	R/W	676
MAL0_RXDEIR	DCR	MAL Rx Descriptor Error Interrupt Register	0x0193	R/Clear	674
MAL0_RXEOBISR	DCR	MAL Rx End of Buffer Interrupt Status Register	0x0192	R/Clear	671
MAL0_RXTTATTR	DCR	MAL Rx PLB Attribute Register	0x0194	R/W	675
MAL0_TXBADDR	DCR	MAL Tx Descriptor Base Address Register	0x0189	R/W	676
MAL0_TXCARR	DCR	MAL Tx Channel Active Register (Reset)	0x0185	R/W	670
MAL0_TXCASR	DCR	MAL Tx Channel Active Register (Set)	0x0184	R/W	670
MAL0_TXCTP0R	DCR	MAL Tx Channel 0 Table Pointer Register	0x01A0	R/W	676
MAL0_TXCTP1R	DCR	MAL Tx Channel 1 Table Pointer Register	0x01A1	R/W	676
MAL0_TXDEIR	DCR	MAL Tx Descriptor Error Interrupt Register	0x0187	R/Clear	674

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
MAL0_TXEOBISR	DCR	MAL Tx End of Buffer Interrupt Status Register	0x0186	R/Clear	671
MAL0_TXTATTRR	DCR	MAL Tx PLB Attribute Register	0x0188	R/W	675
NDFC0_ADDR	MMIO	NDFC Address Register	0x1 xxxx 0004	R/W	601
NDFC0_B0CR	MMIO	NDFC Bank Configuration Register 0	0x1 xxxx 0030	R/W	603
NDFC0_B1CR	MMIO	NDFC Bank Configuration Register 1	0x1 xxxx 0034	R/W	603
NDFC0_B2CR	MMIO	NDFC Bank Configuration Register 2	0x1 xxxx 0038	R/W	603
NDFC0_B3CR	MMIO	NDFC Bank Configuration Register 3	0x1 xxxx 003C	R/W	603
NDFC0_CMD	MMIO	NDFC Command Register	0x1 xxxx 0000	R/W	601
NDFC0_CR	MMIO	NDFC Configuration Register	0x1 xxxx 0040	R/W	604
NDFC0_DATA	MMIO	NDFC Data Register	0x1 xxxx 0008	R/W	601
NDFC0_ECC0	MMIO	NDFC ECC Register 0	0x1 xxxx 0010	R	603
NDFC0_ECC1	MMIO	NDFC ECC Register 1	0x1 xxxx 0014	R	603
NDFC0_ECC2	MMIO	NDFC ECC Register 2	0x1 xxxx 0018	R	603
NDFC0_ECC3	MMIO	NDFC ECC Register 3	0x1 xxxx 001C	R	603
NDFC0_ECC4	MMIO	NDFC ECC Register 4	0x1 xxxx 0020	R	603
NDFC0_ECC5	MMIO	NDFC ECC Register 5	0x1 xxxx 0024	R	603
NDFC0_ECC6	MMIO	NDFC ECC Register 6	0x1 xxxx 0028	R	603
NDFC0_ECC7	MMIO	NDFC ECC Register 7	0x1 xxxx 002C	R	603
NDFC0_HWCTL	MMIO	NDFC Direct Hardware Control Register	0x1 xxxx 0048	R/W	607
NDFC0_REVID	MMIO	NDFC Revision ID Register	0x1 xxxx 0050	R	608
NDFC0_SR	MMIO	NDFC Status Register	0x1 xxxx 0044	R	606
OHCI0_HCREV	MMIO	Revision *(PPC440EPx only)*	0x0 E000 0400	R	818
OHCI0_HCCTRL	MMIO	Control *(PPC440EPx only)*	0x0 E000 0404	R/W	818
OHCI0_HCCMDSTS	MMIO	Command Status *(PPC440EPx only)*	0x0 E000 0408	See register	818
OHCI0_HCINTSTS	MMIO	Interrupt Status *(PPC440EPx only)*	0x0 E000 040C	R/W	818
OHCI0_HCINTE	MMIO	Interrupt Enable *(PPC440EPx only)*	0x0 E000 0410	R/W	818
OHCI0_HCINTDIS	MMIO	Interrupt Disable *(PPC440EPx only)*	0x0 E000 0414	R/W	818
OHCI0_HCHCCA	MMIO	Host Controller Communications Area *(PPC440EPx only)*	0x0 E000 0418	R/W	818
OHCI0_HCPCED	MMIO	Periodic Current Endpoint Descriptor *(PPC440EPx only)*	0x0 E000 041C	R/W	818
OHCI0_HCCHED	MMIO	Control Head Endpoint Descriptor *(PPC440EPx only)*	0x0 E000 0420	R/W	818
OHCI0_HCCTRLCED	MMIO	Control Current Endpoint Descriptor *(PPC440EPx only)*	0x0 E000 0424	R/W	818
OHCI0_HCBULKHED	MMIO	Bulk Head Endpoint Descriptor *(PPC440EPx only)*	0x0 E000 0428	R/W	818
OHCI0_HCBULKCED	MMIO	Bulk Current Endpoint Descriptor *(PPC440EPx only)*	0x0 E000 042C	R/W	818
OHCI0_HCDHEAD	MMIO	Done Head Transfer Descriptor *(PPC440EPx only)*	0x0 E000 0430	R	818
OHCI0_HCFMINT	MMIO	Frame Interval *(PPC440EPx only)*	0x0 E000 0434	R/W	818
OHCI0_HCFMREM	MMIO	Frame Remaining *(PPC440EPx only)*	0x0 E000 0438	R	818
OHCI0_HCFMNUM	MMIO	Frame Number *(PPC440EPx only)*	0x0 E000 043C	R	818
OHCI0_HCPRDSTRT	MMIO	Periodic Start *(PPC440EPx only)*	0x0 E000 0440	R/W	818

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
OHCI0_HCLSTHRES	MMIO	Low Speed Threahold *(PPC440EPx only)*	0x0 E000 0444	R/W	818
OHCI0_HCRHDESCA	MMIO	Root Hub Descriptor A *(PPC440EPx only)*	0x0 E000 0448	See register	818
OHCI0_HCRHDESCB	MMIO	Root Hub Descriptor B *(PPC440EPx only)*	0x0 E000 044C	R/W	818
OHCI0_HCRHSTS	MMIO	Root Hub Status *(PPC440EPx only)*	0x0 E000 0450	See register	818
OHCI0_HCPRSTS	MMIO	Root Hub Port Status *(PPC440EPx only)*	0x0 E000 0454	R/W	818
OPBA0_CR	MMIO	OPB 0 Arbiter Control Register *(PPC440EPx only)*	0x1 EF60 0A01	R/W	104
OPBA0_PR	MMIO	OPB 0 Arbiter Priority Register *(PPC440EPx only)*	0x1 EF60 0A00	R/W	103
OPBA1_CR	MMIO	OPB 1 Arbiter Control Register *(PPC440EPx only)*	0x0 E000 0001	R/W	103
OPBA1_PR	MMIO	OPB 1 Arbiter Priority Register *(PPC440EPx only)*	0x0 E000 0000	R/W	103
OPBA2_CR	MMIO	OPB 2 Arbiter Control Register *(PPC440EPx only)*	0x0 E000 0201	R/W	103
OPBA2_PR	MMIO	OPB 2 Arbiter Priority Register *(PPC440EPx only)*	0x0 E000 0200	R/W	103
OPB2PLB40_BCTRL	DCR	OPB to PLB 4 Bridge Control Register *(PPC440EPx only)*	0x0350	R/W	105
OPB2PLB40_BSTAT	DCR	OPB to PLB 4 Bridge Status Register *(PPC440EPx only)*	0x0351	R	105
OPB2PLB40_BEARL	DCR	OPB-to-PLB4 Error Address Register Low *(PPC440EPx only)*	0x0352	R	106
OPB2PLB40_BEARH	DCR	OPB-to-PLB4 Error Address Register High *(PPC440EPx only)*	0x0353	R	106
OPB2PLB40_REVID	DCR	OPB to PLB 4 Bridge Revision ID Register *(PPC440EPx only)*	0x0354	R	106
P3P4BI0_BEARH	DCR	PLB3 to PLB4 Bridge Error Address High	0x0033	R/Clear	85
P3P4BI0_BEARL	DCR	PLB3 to PLB4 Bridge Error Address Low	0x0032	R/Clear	85
P3P4BI0_BESR0	DCR	PLB3 to PLB4 Bridge Error Status 0	0x0030	R/Clear	85
P3P4BI0_BESR1	DCR	PLB3 to PLB4 Bridge Error Status 1	0x0034	R/Clear	87
P3P4BI0_CFG	DCR	PLB3 to PLB4 Bridge Configuration	0x0036	R/Clear/Set	88
P3P4BI0_PEIR	DCR	PLB3 to PLB4 Bridge Parity Error Interrupt	0x0038	R/Clear/Set	90
P3P4BI0_PICR	DCR	PLB3 to PLB4 Bridge Priority	0x0037	R/Clear/Set	89
P3P4BI0_REVID	DCR	PLB3 to PLB4 Bridge Revision ID	0x003A	R/Clear	90
P4P3BO0_BEARH	DCR	PLB4 to PLB3 Bridge Error Address High	0x0023	R/Clear	77
P4P3BO0_BEARL	DCR	PLB4 to PLB3 Bridge Error Address Low	0x0022	R/Clear	77
P4P3BO0_BESR0	DCR	PLB4 to PLB3 Bridge Error Status 0	0x0020	R/Clear	77
P4P3BO0_BESR1	DCR	PLB4 to PLB3 Bridge Error Status 1	0x0024	R/Clear	80
P4P3BO0_CFG	DCR	PLB4 to PLB3 Bridge Configuration	0x0026	R/Clear/Set	82
P4P3BO0_PEIR	DCR	PLB4 to PLB3 Bridge Parity Error Interrupt	0x0028	R/Clear/Set	84
P4P3BO0_PICR	DCR	PLB4 to PLB3 Bridge Priority	0x0027	R/Clear/Set	83
P4P3BO0_REVID	DCR	PLB4 to PLB3 Bridge Revision ID	0x002A	R/Clear	84
PCIC0_BIST	MMIO	PCI Built In Self Test Control	offset 0x8000 000F	R	482
PCIC0_BRDGOPT1	MMIO	PCI Bridge Options 1	offset 0x8000 004A	R/W	487
PCIC0_BRDGOPT2	MMIO	PCI Bridge Options 2	offset 0x8000 0060	R/W	492
PCIC0_CACHELS	MMIO	PCI Cache Line Size	offset 0x8000 000C	R	482
PCIC0_CAP	MMIO	PCI Capabilities Pointer	offset 0x8000 0034	R	484
PCIC0_CAPID	MMIO	Capability Identifier	offset 0x8000 0058	R	490

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PCIC0_CFGADDR	MMIO	PCI Space Configuration Address	0x1 EEC0 0000	R/W	478
PCIC0_CFGDATA	MMIO	PCI Space Configuration Data	0x1 EEC0 0004	R/W	478
PCIC0_CLS	MMIO	PCI Class	offset 0x8000 0009	R/W	481
PCIC0_CMD	MMIO	PCI Command	offset 0x8000 0004	R/W	479
PCIC0_DATA	MMIO	Data	offset 0x8000 005F	R	492
PCIC0_DEVID	MMIO	PCI Device ID	offset 0x8000 0002	R/W	479
PCIC0_ERREN	MMIO	Error Enable	offset 0x8000 0048	R/W	485
PCIC0_ERRSTS	MMIO	Error Status	offset 0x8000 0049	R/W	486
PCIC0_HDTYPE	MMIO	PCI Header Type	offset 0x8000 000E	R	482
PCIC0_ICS	MMIO	PCI Interrupt Control/Status	offset 0x8000 0044	R/W	485
PCIC0_INTLN	MMIO	PCI Interrupt Line	offset 0x8000 003C	R/W	484
PCIC0_INTPN	MMIO	PCI Interrupt Pin	offset 0x8000 003D	R	485
PCIC0_LATTIM	MMIO	PCI Latency Timer	offset 0x8000 000D	R/W	482
PCIC0_MAXLTNCY	MMIO	PCI Maximum Latency	offset 0x8000 003F	R	485
PCIC0_MINGNT	MMIO	PCI Minimum Grant	offset 0x8000 003E	R	485
PCIC0_NEXTIPTR	MMIO	Next Item Pointer	offset 0x8000 0059	R	490
PCIC0_PLBBEAR	MMIO	PLB Slave Error Address Register	offset 0x8000 0054	R/W	490
PCIC0_PLBBESR0	MMIO	PLB Slave Error Syndrome 0	offset 0x8000 004C	R/W	487
PCIC0_PLBBESR1	MMIO	PLB Slave Error Syndrome 1	offset 0x8000 0050	R/W	489
PCIC0_PMC	MMIO	Power Management Capabilities	offset 0x8000 005A	R	491
PCIC0_PMCSR	MMIO	Power Management Control Status	offset 0x8000 005C	R/W	491
PCIC0_PMCSRBSE	MMIO	PMCSR PCI-to-PCI Bridge Support Extensions	offset 0x8000 005E	R	492
PCIC0_PMSCRR	MMIO	Power Management State Change Request Register	offset 0x8000 0064	R/W	493
PCIC0_PTM1BAR	MMIO	PCI PTM 1 BAR	offset 0x8000 0014	R/W	483
PCIC0_PTM2BAR	MMIO	PCI PTM 2 BAR	offset 0x8000 0018	R/W	483
PCIC0_REVID	MMIO	PCI Revision ID	offset 0x8000 0008	R/W	481
PCIC0_SBSYSID	MMIO	PCI Subsystem ID	offset 0x8000 002E	R/W	484
PCIC0_SBSYSVID	MMIO	PCI Subsystem Vendor ID	offset 0x8000 002C	R/W	484
PCIC0_STATUS	MMIO	PCI Status	offset 0x8000 0006	R/W	480
PCIC0_VENDID	MMIO	PCI Vendor ID	offset 0x8000 0000	R/W	479
PCIL0_PMM0LA	MMIO	PMM 0 Local Address	0x1 EF40 0000	R/W	473
PCIL0_PMM0MA	MMIO	PMM 0 Mask/Attribute	0x1 EF40 0004	R/W	473
PCIL0_PMM0PCIHA	MMIO	PMM 0 PCI High Address	0x1 EF40 000C	R/W	474
PCIL0_PMM0PCILA	MMIO	PMM 0 PCI Low Address	0x1 EF40 0008	R/W	474
PCIL0_PMM1LA	MMIO	PMM 1 Local Address	0x1 EF40 0010	R/W	474
PCIL0_PMM1MA	MMIO	PMM 1 Mask/Attribute	0x1 EF40 0014	R/W	474
PCIL0_PMM1PCIHA	MMIO	PMM 1 PCI High Address	0x1 EF40 001C	R/W	475
PCIL0_PMM1PCILA	MMIO	PMM 1 PCI Low Address	0x1 EF40 0018	R/W	475

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PCIL0_PMM2LA	MMIO	PMM 2 Local Address	0x1 EF40 0020	R/W	475
PCIL0_PMM2MA	MMIO	PMM 2 Mask/Attribute	0x1 EF40 0024	R/W	475
PCIL0_PMM2PCIHA	MMIO	PMM 2 PCI High Address	0x1 EF40 002C	R/W	476
PCIL0_PMM2PCILA	MMIO	PMM 2 PCI Low Address	0x1 EF40 0028	R/W	476
PCIL0_PTM1LA	MMIO	PTM 1 Local Address	0x1 EF40 0034	R/W	476
PCIL0_PTM1MS	MMIO	PTM 1 Memory Size/Attribute	0x1 EF40 0030	R/W	476
PCIL0_PTM2LA	MMIO	PTM 2 Local Address	0x1 EF40 003C	R/W	477
PCIL0_PTM2MS	MMIO	PTM 2 Memory Size/Attribute	0x1 EF40 0038	R/W	477
PLB32OPB0_BEAR	DCR	PLB 3 to OPB 0 Bridge Error Address Register	0x0092	R	101
PLB32OPB0_BESR0	DCR	PLB 3 to OPB 0 Bridge Error Status Register 0	0x0090	R/Clear	100
PLB32OPB0_BESR1	DCR	PLB 3 to OPB 0 Bridge Error Status Register 1	0x0094	R/Clear	102
PLB32OPB0_REVID	DCR	PLB 3 to OPB 0 Bridge Revision ID Register	0x0093	R	101
PLB3A0_ACR	DCR	PLB3 Arbiter Control Register	0x0077	R/W	98
PLB3A0_BEAR	DCR	PLB3 Arbiter Bus Error Address Register	0x0076	R/W	98
PLB3A0_BESR	DCR	PLB3 Arbiter Bus Error Status Register	0x0074	R/Clear	97
PLB3A0_REVID	DCR	PLB3 Arbiter Revision ID Register	0x0072	R	96
PLB42OPB0_BEARH	DCR	PLB 4 to OPB 0 Bridge Error Address Register High	0x0203	R	93
PLB42OPB0_BEARL	DCR	PLB 4 to OPB 0 Bridge Error Address Register Low	0x0202	R	93
PLB42OPB0_BESR0	DCR	PLB 4 to OPB 0 Bridge Error Status Register 0	0x0200	R/Clear	91
PLB42OPB0_BESR1	DCR	PLB 4 to OPB 0 Bridge Error Status Register 1	0x0200	R/Clear	94
PLB42OPB0_CFG	DCR	PLB 4 to OPB 0 Bridge Configuration Register	0x0206	R/Clear	95
PLB42OPB0_LATENCY	DCR	PLB 4 to OPB 0 Bridge Burst Latency Register	0x0208	R/Clear	95
PLB42OPB0_REVID	DCR	PLB 4 to OPB 0 Bridge Revision ID Register	0x020A	R	96
PLB42OPB1_BEARH	DCR	PLB 4 to OPB 1 Bridge Error Address Register High	0x0343	R	93
PLB42OPB1_BEARL	DCR	PLB 4 to OPB 1 Bridge Error Address Register Low	0x0342	R	93
PLB42OPB1_BESR0	DCR	PLB 4 to OPB 1 Bridge Error Status Register 0	0x0340	R/Clear	91
PLB42OPB1_BESR1	DCR	PLB 4 to OPB 1 Bridge Error Status Register 1	0x0344	R/Clear	94
PLB42OPB1_CFG	DCR	PLB 4 to OPB 1 Bridge Configuration Register	0x0346	R/Clear	95
PLB42OPB1_LATENCY	DCR	PLB 4 to OPB 1 Bridge Burst Latency Register	0x0348	R/Clear	95
PLB42OPB1_REVID	DCR	PLB 4 to OPB 1 Bridge Revision ID Register	0x034A	R	96
PLB4A0_ACR	DCR	PLB4A0 Arbiter Control Register	0x0081	R/W	70
PLB4A0_CCR	DCR	PLB4 Crossbar Control Register	0x0088	R/W	75
PLB4A0_EARH	DCR	PLB4A0 Error Address Register High	0x0085	R	75
PLB4A0_EARL	DCR	PLB4A0 Error Address Register Low	0x0084	R	75
PLB4A0_ESRH*	DCR	PLB4A0 Error Status Register High (*diagnostic use only)	0x0087	Set(W)	na
PLB4A0_ESRL	DCR	PLB4A0 Error Status Register Low	0x0082	R/Clear	73
PLB4A0_ESRL*	DCR	PLB4A0 Error Status Register Low (*diagnostic use only)	0x0086	Set(W)	73

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
PLB4A0_REVID	DCR	PLB4 Crossbar ID/Revision Register	0x0080	R	69
PLB4A1_ACR	DCR	PLB4A1 Arbiter Control Register	0x0089	R/W	70
PLB4A1_EARH	DCR	PLB4A1 Error Address Register High	0x008D	R	75
PLB4A1_EARL	DCR	PLB4A1 Error Address Register Low	0x008C	R	75
PLB4A1_ESRH*	DCR	PLB4A1 Error Status Register High (*diagnostic use only)	0x008F	Set(W)	na
PLB4A1_ESRL	DCR	PLB4A1 Error Status Register Low	0x008A	R/Clear	73
PLB4A1_ESRL*	DCR	PLB4A1 Error Status Register Low (*diagnostic use only)	0x008E	Set(W)	73
RGMII0_FER	MMIO	RGMII Function Enable Register	0x1 EF601000	R/W	690
RGMII0_SSR	MMIO	RGMII Speed Select Register	0x1 EF601004	R/W	690
SDR0_AMP0	DCR	Alternate PLB4 Master Priority Register	offset 0x0240	R/W	60
SDR0_AMP1	DCR	Alternate PLB3 Master Priority Register	offset 0x0241	R/W	62
SDR0_CFGADDR	DCR	System DCR Configuration Address Register	0x000E	R/W	176
SDR0_CFGDATA	DCR	System DCR Configuration Data Register	0x000F	R/W	176
SDR0_CP440	DCR	440CPU Control Register	offset 0x0180	R/W	63
SDR0_CRYP0	DCR	Security Function Configuration Register	offset 0x4500	R/W	312
SDR0_CUST0	DCR	Register0 Reserved for Customer Use	offset 0x4000	R/W	203
SDR0_CUST1	DCR	Register1 Reserved for Customer Use	offset 0x4002	R/W	204
SDR0_DDRCFG	DCR	DDR Configuration Register	offset 0x00E0	R/W	182
SDR0_EBC0	DCR	EBC Configuration Register	offset 0x0100	R/W	204
SDR0_ECID0	DCR	Electronic Chip ID Register 0	offset 0x0080	R/W	177
SDR0_ECID1	DCR	Electronic Chip ID Register 1	offset 0x0081	R/W	177
SDR0_ECID2	DCR	Electronic Chip ID Register 2	offset 0x0082	R/W	177
SDR0_ECID3	DCR	Electronic Chip ID Register 3	offset 0x0083	R/W	177
SDR0_EMAC0REJCNT	DCR	EMAC0 RX Packet Reject Counter	offset 0x4305	R	702
SDR0_EMAC0RXST	DCR	EMAC0 RX Status Register	offset 0x4301	R/W	699
SDR0_EMAC0TXST	DCR	EMAC0 TX Status Register	offset 0x4302	R/W	701
SDR0_EMAC1REJCNT	DCR	EMAC1 RX Packet Reject Counter	offset 0x4306	R	702
SDR0_EMAC1RXST	DCR	EMAC1 RX Status Register	offset 0x4303	R/W	699
SDR0_EMAC1TXST	DCR	EMAC1 TX Status Register	offset 0x4304	R/W	701
SDR0_ICCRTX	DCR	Interrupt Coalescing Control Transmit Register	offset 0x430B	R/W	678
SDR0_ICCRRX	DCR	Interrupt Coalescing Control Receive Register	offset 0x430C	R/W	679
SDR0_ICTRTX0	DCR	Interrupt Coalescing Timer Register Transmit Channel 0	offset 0x430D	R/W	680
SDR0_ICTRTX1	DCR	Interrupt Coalescing Timer Register Transmit Channel 1	offset 0x430E	R/W	680
SDR0_ICTRRX0	DCR	Interrupt Coalescing Timer Register Receive Channel 0	offset 0x430F	R/W	680
SDR0_ICTRRX1	DCR	Interrupt Coalescing Timer Register Receive Channel 1	offset 0x4310	R/W	680
SDR0_ICSRTX0	DCR	Interrupt Coalescing Status Register Transmit Channel 0	offset 0x4307	R/Clear	680
SDR0_ICSRTX1	DCR	Interrupt Coalescing Status Register Transmit Channel 1	offset 0x4308	R/Clear	680
SDR0_ICSRRX0	DCR	Interrupt Coalescing Status Register Receive Channel 0	offset 0x4309	R/Clear	680

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
SDR0_ICSRRX1	DCR	Interrupt Coalescing Status Register Receive Channel 1	offset 0x430A	R/Clear	680
SDR0_JTAG	DCR	JTAG ID Register	offset 0x00C0	R/W	na[1]
SDR0_MALRBL	DCR	MAL Receive Burst Length Register	offset 0x02A0	R/W	651
SDR0_MALRBS	DCR	MAL Receive Bus Size Register	offset 0x02E0	R/W	651
SDR0_MALTBL	DCR	MAL Transmit Burst Length Register	offset 0x0280	R/W	652
SDR0_MALTBS	DCR	MAL Transmit Bus Size Register	offset 0x02C0	R/W	652
SDR0_MFR	DCR	Miscellaneous Function Register	offset 0x4300	R/W	181
SDR0_MIRQ0	DCR	Master Interrupt Request Register 0	offset 0x0260	R/W	64
SDR0_MIRQ1	DCR	Master Interrupt Request Register 1	offset 0x0261	R/W	64
SDR0_PCI0	DCR	PCI Control Register	offset 0x0300	R/W	470
SDR0_PFC0	DCR	Pin Function Control Register 0	offset 0x4100	R/W	177
SDR0_PFC1	DCR	Pin Function Control Register 1	offset 0x4101	R/W	178
SDR0_PFC2	DCR	Ethernet PLL Configuration Register	offset 0x4102	R/W	691
SDR0_PFC4	DCR	GPIO Multiplexing Register	offset 0x4104	R/W	179
SDR0_PINSTP	DCR	Pin Strapping Register	offset 0x0040	R	197
SDR0_SDCS0	DCR	Serial Device Controller Settings Register	offset 0x0060	R/W	197
SDR0_SDSTP0	DCR	Serial Device Strap Register 0	offset 0x0020	R	198
SDR0_SDSTP1	DCR	Serial Device Strap Register 1	offset 0x0021	R	200
SDR0_SDSTP2	DCR	Read Only Version of SDR0_CUST0	offset 0x4001	R	201
SDR0_SDSTP3	DCR	Read Only Version of SDR0_CUST1	offset 0x4003	R	202
SDR0_SLPIPE0	DCR	PLB Slave Address Pipeline Disabling Register	offset 0x0220	R/W	67
SDR0_SRST0	DCR	Individual Core Reset Control Register 0	offset 0x0200	R/W	179
SDR0_SRST1	DCR	Individual Core Reset Control Register 1	offset 0x0201	R/W	179
SDR0_UART0	DCR	UART Configuration Register 0	offset 0x0120	R/W	754
SDR0_UART1	DCR	UART Configuration Register 1	offset 0x0121	R/W	755
SDR0_UART2	DCR	UART Configuration Register 2	offset 0x0122	R/W	756
SDR0_UART3	DCR	UART Configuration Register 3	offset 0x0123	R/W	756
SDR0_USB2D0CR	DCR	USB2 Device Control Register *(PPC440EPx only)*	offset 0x0320	R/W	812
SDR0_USB2H0CR	DCR	USB2 Host Control Register *(PPC440EPx only)*	offset 0x0340	R/W	811
SDR0_USB2PHY0CR	DCR	USB2PHY0 Control Register *(PPC440EPx only)*	offset 0x4103	R/W	810
SDR0_USB2H0ST	DCR	USB2H0 Status Register *(PPC440EPx only)*	offset 0x4600	R	811
SPI0_CDM	MMIO	SPI Clock Divisor Modulus Register	0x1 EF60 0906	R/W	773
SPI0_CR	MMIO	SPI Control Register	0x1 EF60 0903	R/W	772
SPI0_MODE	MMIO	SPI Mode Register	0x1 EF60 0900	R/W	774
SPI0_RxD	MMIO	SPI Receive Data Register	0x1 EF60 0901	R	772
SPI0_SR	MMIO	SPI Status Register	0x1 EF60 0904	R	773
SRAM0_BEAR	DCR	SRAM Bus Error Address Register	0x0384	R/W	448
SRAM0_BESR0	DCR	Bus Error Status Register 0	0x0385	R/W	449

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
SRAM0_BESR1	DCR	Bus Error Status Register 1	0x0386	R/W	450
SRAM0_CID	DCR	Core ID Register	0x0388	R	452
SRAM0_DPC	DCR	Data Parity Check	0x038A	R/W	453
SRAM0_PMEG	DCR	Power Management	0x0387	R/W	452
SRAM0_REVID	DCR	Revision ID	0x0389	R	453
SRAM0_SB0CR	DCR	SRAM Bank 0 Configuration Register	0x0380	R/W	448
UART0_DLL	MMIO	UART 0 Baud-rate Divisor Latch LSB	0x1 EF60 0300	R/W	753
UART0_DLM	MMIO	UART 0 Baud-rate Divisor Latch MSB	0x1 EF60 0301	R/W	753
UART0_FCR	MMIO	UART 0 FIFO Control Register	0x1 EF60 0302	W	749
UART0_IER	MMIO	UART 0 Interrupt Enable Register	0x1 EF60 0301	R/W	747
UART0_IIR	MMIO	UART 0 Interrupt Identification Register	0x1 EF60 0302	R	748
UART0_LCR	MMIO	UART 0 Line Control Register	0x1 EF60 0303	R/W	749
UART0_LSR	MMIO	UART 0 Line Status Register	0x1 EF60 0305	R/W	751
UART0_MCR	MMIO	UART 0 Modem Control Register	0x1 EF60 0304	R/W	750
UART0_MSR	MMIO	UART 0 Modem Status Register	0x1 EF60 0306	R/W	752
UART0_RBR	MMIO	UART 0 Receiver Buffer Register	0x1 EF60 0300	R	747
UART0_SCR	MMIO	UART 0 Scratch Register	0x1 EF60 0307	R/W	753
UART0_THR	MMIO	UART 0 Transmitter Holding Register	0x1 EF60 0300	W	747
UART1_DLL	MMIO	UART 1 Baud-rate Divisor Latch LSB	0x1 EF60 0400	R/W	753
UART1_DLM	MMIO	UART 1 Baud-rate Divisor Latch MSB	0x1 EF60 0401	R/W	753
UART1_FCR	MMIO	UART 1 FIFO Control Register	0x1 EF60 0402	W	749
UART1_IER	MMIO	UART 1 Interrupt Enable Register	0x1 EF60 0401	R/W	747
UART1_IIR	MMIO	UART 1 Interrupt Identification Register	0x1 EF60 0402	R	748
UART1_LCR	MMIO	UART 1 Line Control Register	0x1 EF60 0403	R/W	749
UART1_LSR	MMIO	UART 1 Line Status Register	0x1 EF60 0405	R/W	751
UART1_MCR	MMIO	UART 1 Modem Control Register	0x1 EF60 0404	R/W	750
UART1_MSR	MMIO	UART 1 Modem Status Register	0x1 EF60 0406	R/W	752
UART1_RBR	MMIO	UART 1 Receiver Buffer Register	0x1 EF60 0400	R	747
UART1_SCR	MMIO	UART 1 Scratch Register	0x1 EF60 0407	R/W	753
UART1_THR	MMIO	UART 1 Transmitter Holding Register	0x1 EF60 0400	W	747
UART2_DLL	MMIO	UART 2 Baud-rate Divisor Latch LSB	0x1 EF60 0500	R/W	753
UART2_DLM	MMIO	UART 2 Baud-rate Divisor Latch MSB	0x1 EF60 0501	R/W	753
UART2_FCR	MMIO	UART 2 FIFO Control Register	0x1 EF60 0502	W	749
UART2_IER	MMIO	UART 2 Interrupt Enable Register	0x1 EF60 0501	R/W	747
UART2_IIR	MMIO	UART 2 Interrupt Identification Register	0x1 EF60 0502	R	748
UART2_LCR	MMIO	UART 2 Line Control Register	0x1 EF60 0503	R/W	749
UART2_LSR	MMIO	UART 2 Line Status Register	0x1 EF60 0505	R/W	751
UART2_MCR	MMIO	UART 2 Modem Control Register	0x1 EF60 0504	R/W	750

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
UART2_MSR	MMIO	UART 2 Modem Status Register	0x1 EF60 0506	R/W	752
UART2_RBR	MMIO	UART 2 Receiver Buffer Register	0x1 EF60 0500	R	747
UART2_SCR	MMIO	UART 2 Scratch Register	0x1 EF60 0507	R/W	753
UART2_THR	MMIO	UART 2 Transmitter Holding Register	0x1 EF60 0500	W	747
UART3_DLL	MMIO	UART 3 Baud-rate Divisor Latch LSB	0x1 EF60 0600	R/W	753
UART3_DLM	MMIO	UART 3 Baud-rate Divisor Latch MSB	0x1 EF60 0601	R/W	753
UART3_FCR	MMIO	UART 3 FIFO Control Register	0x1 EF60 0602	W	749
UART3_IER	MMIO	UART 3 Interrupt Enable Register	0x1 EF60 0601	R/W	747
UART3_IIR	MMIO	UART 3 Interrupt Identification Register	0x1 EF60 0602	R	748
UART3_LCR	MMIO	UART 3 Line Control Register	0x1 EF60 0603	R/W	749
UART3_LSR	MMIO	UART 3 Line Status Register	0x1 EF60 0605	R/W	751
UART3_MCR	MMIO	UART 3 Modem Control Register	0x1 EF60 0604	R/W	750
UART3_MSR	MMIO	UART 3 Modem Status Register	0x1 EF60 0606	R/W	752
UART3_RBR	MMIO	UART 3 Receiver Buffer Register	0x1 EF60 0600	R	747
UART3_SCR	MMIO	UART 3 Scratch Register	0x1 EF60 0607	R/W	753
UART3_THR	MMIO	UART 3 Transmitter Holding Register	0x1 EF60 0600	W	747
UIC0_CR	DCR	UIC 0 Critical Register	0x00C3	R/W	221
UIC0_ER	DCR	UIC 0 Enable Register	0x00C2	R/W	215
UIC0_MSR	DCR	UIC 0 Masked Status Register	0x00C6	R	237
UIC0_PR	DCR	UIC 0 Polarity Register	0x00C4	R/W	226
UIC0_SR	DCR	UIC 0 Status Register	0x00C0	R/Clear	210
UIC0_SRS	DCR	UIC 0 Status Register Set (reserved for debug only)	0x00C1	W/Set	na
UIC0_TR	DCR	UIC 0 Triggering Register	0x00C5	R/W	232
UIC0_VCR	DCR	UIC 0 Vector Configuration Register	0x00C8	W	243
UIC0_VR	DCR	UIC 0 Vector Register	0x00C7	R	244
UIC1_CR	DCR	UIC 1 Critical Register	0x00D3	R/W	223
UIC1_ER	DCR	UIC 1 Enable Register	0x00D2	R/W	218
UIC1_MSR	DCR	UIC 1 Masked Status Register	0x00D6	R	240
UIC1_PR	DCR	UIC 1 Polarity Register	0x00D4	R/W	229
UIC1_SR	DCR	UIC 1 Status Register	0x00D0	R/Clear	212
UIC1_SRS	DCR	UIC 1 Status Register Set (reserved for debug only)	0x00D1	W/Set	na
UIC1_TR	DCR	UIC 1 Triggering Register	0x00D5	R/W	234
UIC1_VCR	DCR	UIC 1 Vector Configuration Register	0x00D8	W	243
UIC1_VR	DCR	UIC 1 Vector Register	0x00D7	R	245
UIC2_CR	DCR	UIC 2 Critical Register	0x00E3	R/W	225
UIC2_ER	DCR	UIC 2 Enable Register	0x00E2	R/W	220
UIC2_MSR	DCR	UIC 2 Masked Status Register	0x00E6	R	242
UIC2_PR	DCR	UIC 2 Polarity Register	0x00E4	R/W	231

Table 35-1. Alphabetical Listing of Chip Control and Peripheral Function Registers (continued)

Register	Type	Description	Address or Number	Access	See Page
UIC2_SR	DCR	UIC 2 Status Register	0x00E0	R/Clear	214
UIC2_SRS	DCR	UIC 2 Status Register Set (reserved for debug only)	0x00E1	W/Set	na
UIC2_TR	DCR	UIC 2 Triggering Register	0x00E5	R/W	237
USB2D0_INTRIN	MMIO	Interrupt register for Endpoint 0 plus IN Endpoints 1 to 3 *(PPC440EPx only)*	0x0 E000 0100	R	866
USB2D0_POWER	MMIO	Power Management Register *(PPC440EPx only)*	0x0 E000 0102	R/W	866
USB2D0_FADDR	MMIO	Function Address Register *(PPC440EPx only)*	0x0 E000 0103	See register	866
USB2D0_INTRINE	MMIO	Interrupt enable register for USB2D0_INTRIN *(PPC440EPx only)*	0x0 E000 0104	R/W	866
USB2D0_INTROUT	MMIO	Interrupt register for OUT Endpoints 1 to 3 *(PPC440EPx only)*	0x0 E000 0106	R	866
USB2D0_INTRUSBE	MMIO	Interrupt enable register for USB2D0_INTRUSB *(PPC440EPx only)*	0x0 E000 0108	R/W	866
USB2D0_INTRUSB	MMIO	Interrupt register for common USB interrupts *(PPC440EPx only)*	0x0 E000 0109	R	866
USB2D0_INTROUTE	MMIO	Interrupt enable register for IntrOut *(PPC440EPx only)*	0x0 E000 010A	R/W	866
USB2D0_OUTCSR	MMIO	Control/Status register for OUT Endpoint *(PPC440EPx only)*	0x0 E000 0114	R/W	881
USB2D0_OUTCOUNT0	MMIO	Number of received bytes in Endpoint 0 FIFO *(PPC440EPx only)*	See register	R	884
USB2D0_OUTCOUNT	MMIO	Number of bytes in OUT Endpoint FIFO *(PPC440EPx only)*	0x0 E000 011A	R	884
USB2D0_OUTMAXP	MMIO	Maximum packet size for OUT Endpoint *(PPC440EPx only)*	0x0 E000 0116	R/W	883
USB2D0_TSTMODE	MMIO	Enables the USB 2.0 test modes *(PPC440EPx only)*	0x0 E000 010C	R/W	866
USB2D0_INDEX	MMIO	Index for selecting the Endpoint status/control registers *(PPC440EPx only)*	0x0 E000 010D	R/W	866
USB2D0_FRAME	MMIO	Frame number *(PPC440EPx only)*	0x0 E000 010E	R	866
USB2HMDCR0_BCNTL	DCR	OPB Master Interface Bridge Control *(PPC440EPx only)*	0x0360	R/W	858
USB2HMDCR0_BSTAT	DCR	OPB Master Interface Bridge Status *(PPC440EPx only)*	0x0361	R/W	859
USB2HMDCR0_LAT	DCR	OPB Master Interface Bridge Latency *(PPC440EPx only)*	0x0366	R/W	860
USB2HMDCR0_REVID	DCR	OPB Master Interface Bridge Revision ID *(PPC440EPx only)*	0x0367	R	861
USB2HSDCR_BCNTL	DCR	OPB Slave Interface Bridge Control *(PPC440EPx only)*	0x0370	R/W	862
USB2HSDCR0_REVID	DCR	OPB Slave Interface Bridge Revision ID *(PPC440EPx only)*	0x0373	R	863
ZMII0_FER	MMIO	ZMII Function Enable Register	0x1 EF60 0D00	R/W	685
ZMII0_SMIISR	MMIO	ZMII SMII Status Register	0x1 EF60 0D08	R/W	686
ZMII0_SSR	MMIO	ZMII Speed Select Register	0x1 EF60 0D04	R/W	685

Note 1. Refer to the *PPC440 Processor User's Manual*.

36. Signal Summary

This chapter provides information about the PPC440EPx/GRx I/O signals.

36.1 Signals Listed Alphabetically

The *Signals Listed Alphabetically* table in the *PPC440EPx/GRx Embedded Processor Data Sheets* provides a complete list of all the PPC440EPx/GRx I/O signals in alphabetical order.

36.2 Signals by Interface Category

The *Signal Functional Description* table in the *PPC440EPx/GRx Embedded Processor Data Sheets* provides a complete list of all the PPC440EPx/GRx I/O signals grouped together by interface category. The functional characteristics of the signal are described.

Appendix A. Floating Point Instruction Summary

This appendix contains PPC440 FPU instructions summarized alphabetically and by opcode.

Instruction Set – Alphabetical lists all PPC440 FPU mnemonics, including extended mnemonics, alphabetically. A short functional description is included for each mnemonic.

Instructions Sorted by Opcode, lists all PPC440 FPU instructions, sorted by primary and secondary opcodes. Extended mnemonics are not included in the opcode list.

Instruction Formats, illustrates the PPC440 FPU instruction forms (allowed arrangements of fields within instructions).

Note: The FPU is supported by the PPC440EPx only.

A.1 Instruction Set – Alphabetical

Table A-1 summarizes the PPC440 FPU instruction set. The instructions are described in detail in *Floating Point Instruction Set* on page 891 which is also alphabetized under the root form. It also describes the instruction operands and notation.

Table A-1. PPC440 FPU Instruction Syntax Summary

Mnemonic	Operands	Function	Other Registers Changed	Page
fabs	FRT, FRB	Place (FRB), with bit 0 set to zero, into FRT.		895
fadd	FRT, FRA, FRB	Add (FRA) to (FRB). Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	896
fadds	FRT, FRA, FRB	Add (FRA) to (FRB). Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	897
fcmpo	BF, FRA, FRB	(FRA) is compared to (FRB). Place result in CR[BF] and FPSCR[FPCC].	CR[BF FPSCR[FPCC, FX, VXSNAN, VXVC]	898
fcmpu	BF, FRA, FRB	(FRA) is compared to (FRB). Place result in CR[BF] and FPSCR[FPCC].	CR[BF] FPSCR[FPCC, FX, VXSNAN]	899
fctiw	FRT, FRB	Convert (FRB) to a 32-bit signed integer. Place result in $FRT_{32:63}$. $FRT_{0:31}$ are undefined.	FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI] FPSCR[FPRF] undefined	900
fctiwz	FRT, FRB	Convert (FRB) to a 32-bit signed integer. Place result in $FRT_{32:63}$. $FRT_{0:31}$ are undefined.	FPSCR[FR, FI, FX, XX, VXSNAN, VXCVI] FPSCR[FPRF] undefined	901
fdiv	FRT, FRA, FRB	Divide (FRA) by (FRB). Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]	902

Table A-1. PPC440 FPU Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
fdivs	FRT, FRA, FRB	Divide (FRA) by (FRB). Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, ZX, XX, VXSNAN, VXIDI, VXZDZ]	903
fmadd	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	904
fmadds	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	905
fmsub	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	907
fmsubs	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI]	908
fmul	FRT, FRA, FRC	Multiply (FRA) by (FRC). Place result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]	909
fmuls	FRT, FRA, FRC	Multiply (FRA) by (FRC). Round result to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXIMZ]	910
fnabs	FRT, FRB	Place (FRB), with bit 0 set to one, into FRT.		911
fneg	FRT, FRB	Place (FRB), with bit 0 inverted, into FRT.		912
fnmadd	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Negate result and place it in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	913
fnmadds	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Add (FRB) to the intermediate result. Negate result, round it to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	905
fnmsub	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Negate result and place it in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	915
fnmsubs	FRT, FRA, FRC, FRB	Multiply (FRA) by (FRC). Subtract (FRB) from the intermediate result. Negate result, round it to single-precision format and place in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN, VXISI, VXIMZ]	916

Table A-1. PPC440 FPU Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
fres	FRT, FRB	A single-precision estimate of the reciprocal of the floating-point operand in FRB is placed into FRT.	FPSCR[FX, OX, UX, ZX, VXSNAN], FPSCR[FPRF, FR, FI] undefined	917
frsp	FRT, FRB	Round (FRB) to single precision and place the result in FRT.	FPSCR[FPRF, FR, FI, FX, OX, UX, XX, VXSNAN]	918
frsqrte	FRT, FRB	A double-precision estimate of the reciprocal of the square root of the floating-point operand in FRB is placed into FRT.	FPSCR[FX, ZX, VXSNAN, VXSQRT], FPSCR[FPRF, FR, FI] undefined	919
fsel	FRT, FRA, FRC, FRB	The floating-point operand in FRA is compared to zero.		921
fsub	FRT, FRA, FRB	The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).		922
fsubs	FRT, FRA, FRB	The floating-point operand in FPR(FRB) is subtracted from the floating-point operand in FPR(FRA).	FPSCR[FPRF, FR, FI FX, OX, UX, XX, VXSNAN]	923
lfd	FRT, D(RA)	Load the double word from EA = (RA|0) + EXTS(D) into FRT.		924
lfdu	FRT, D(RA)	Load the double word from EA = (RA|0) + EXTS(D) into FRT. Update the base address, (RA) ← EA		925
lfdux	FRT, RA, RB	Load the double word from EA = (RA|0) + (RB) into FRT. Update the base address, (RA) ← EA.		926
lfdx	FRT, RA, RB	Load the double word from EA = (RA|0) + (RB) into FRT.		927
lfs	FRT, D(RA)	Load the word from EA = (RA|0) + EXTS(D) into FRT. The word is interpreted as a single precision operand.		928
lfsu	FRT, D(RA)	Load the word from EA = (RA|0) + EXTS(D) into FRT. The word is interpreted as a single precision operand. Update the base address, (RA) ← EA.		929

Table A-1. PPC440 FPU Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lfsux	FRT, RA, RB	Load the word from EA = (RA|0) + (RB) into FRT. The word is interpreted as a single precision operand. Update the base address, (RA) ← EA.		930
lfsx	FRT, RA, RB	Load the word from EA = (RA|0) + (RB). The word is interpreted as a single precision operand.		931
mcrfs	BF, BFA	Copy the contents of the FPSCR specified by BFA to the CR field specified by BF. Place result in CR[BF] and FPSCR[FPCC].	CR[BF See the instruction description for details of changed FPSCR fields	932
mffs	FRT	Place (FPSCR) into $FRT_{32:63}$. $FRT_{0:31}$ are undefined.		933
mtfsb0	BT	Set the FPSCR bit specified by BT to 0.	$FPSCR_{BT}$	934
mtfsb1	BT	Set the FPSCR bit specified by BT to 1.	$FPSCR_{BT}$	935
mtfsf	FLM, FRB	Place $FRB_{32:63}$ in the FPSCR under control of the mask specified by FLM.	$FPSCR_{BF}$	936
mtfsfi	FLM, FRB	Place $FRB_{32:63}$ in the FPSCR under control of the mask specified by FLM.	FPSCR fields altered by mask.	937
stfd	FRS, D(RA)	Store double word (FRS) at EA = (RA|0) + EXTS(D).		938
stfdu	FRS, D(RA)	Store double word (FRS) at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		939
stfdux	FRS, RA, RB	Store double word (FRS) at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		940
stfdx	FRS, RA, RB	Store double word (FRS) at EA = (RA|0) + (RB).		941
stfiwx	FRS, RA, RB	Store word $FRS_{32:63}$ at EA = (RA|0) + (RB).		942
stfs	FRS, D(RA)	Convert (FRS) to single-precision format and store at EA = (RA|0) + EXTS(D).		943
stfsu	FRS, D(RA)	Convert (FRS) to single-precision format and store at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		944
stfsux	FRS, RA, RB	Convert (FRS) to single-precision format and store at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		945
stfsx	RS, RA, RB	Convert (FRS) to single-precision format and store at EA = (RA|0) + (RB).		946

A.2 Instructions Sorted by Opcode

All instructions are four bytes long and word aligned. All instructions have a primary opcode field (shown as field OPCD in *Figure A-1* through *Figure A-10*, beginning on page 981) in bits 0:5. Some instructions also have a secondary opcode field (shown as field XO in *Figure A-1* through *Figure A-10*). PPC440 FPU instructions, sorted by primary and secondary opcode, are listed in *Table A-2*.

The "Form" indicated in the table refers to the arrangement of valid field combinations within the four-byte instruction. See *Instruction Formats* on page 978, for the field layouts of each form.

Form X has a 10-bit secondary opcode field, while form XO uses only the low-order 9-bits of that field. Form XO uses the high-order secondary opcode bit (the tenth bit) as a variable; therefore, every form XO instruction really consumes two secondary opcodes from the 10-bit secondary-opcode space. The implicitly consumed secondary opcode is listed in parentheses for form XO instructions in the table below.

Table A-2. PPC440 FPU Instructions by Opcode

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	535	X	**lfsx**	FRT, RA, RB	931
31	567	X	**lfsux**	FRT, RA, RB	930
31	599	X	**lfdx**	FRT, RA, RB	927
31	631	X	**lfdux**	FRT, RA, RB	926
31	695	X	**stfsux**	FRS, RA, RB	945
31	727	X	**stfdx**	FRS, RA, RB	941
31	759	X	**stfdux**	FRS, RA, RB	940
31	983	X	**stfiwx**	FRS, RA, RB	942
48		D	**lfs**	FRT, D(RA)	928
49		D	**lfsu**	FRT, D(RA)	929
50		D	**lfd**	FRT, D(RA)	924
51		D	**lfdu**	FRT, D(RA)	925
52		D	**stfs**	FRT, D(RA)	943
53		D	**stfsu**	FRT, D(RA)	944
54		D	**stfd**	FRS, D(RA)	938
55		D	**stfdu**	FRS, D(RA)	939
59	18	A	**fdivs**	FRT, FRA, FRB	903
59	20	A	**fsubs**	FRT, FRA, FRB	923
59	21	A	**fadds**	FRT, FRA, FRB	897
59	25	A	**fmuls**	FRT, FRA, FRC	910
59	28	A	**fmsubs**	FRT, FRA, FRC, FRB	908
59	29	A	**fmadds**	FRT, FRA, FRC, FRB	914

Table A-2. PPC440 FPU Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
59	30	A	**fnmsubs**	FRT, FRA, FRC, FRB	916
59	31	A	**fnmadds**	FRT, FRA, FRC, FRB	914
63	12	X	**frsp**	FRT, FRB	918
63	14	X	**fctiw**	FRT, FRB	900
63	15	X	**fctiwz**	FRT, FRB	901
63	18	A	**fdiv**	FRT, FRA, FRB	902
63	20	A	**fsub**	FRT, FRA, FRB	922
63	21	A	**fadd**	FRT, FRA, FRB	896
63	25	A	**fmul**	FRT, FRA, FRC	909
63	28	A	**fmsub**	FRT, FRA, FRC, FRB	907
63	29	A	**fmadd**	FRT, FRA, FRC, FRB	904
63	30	A	**fnmsub**	FRT, FRA, FRC, FRB	915
63	31	A	**fnmadds**	FRT, FRA, FRC, FRB	914
63	38	X	**mtfsb1**	BT	935
63	40	X	**fneg**	FRT, FRB	912
63	70	X	**mtfsb0**	BT	934
63	72	X	**fmr**	FRT, FRB	906
63	134	X	**mtfsfi**	BF, U	937
63	136	X	**fnabs**	FRT, FRB	911
63	264	X	**fabs**	FRT, FRB	895
63	583	X	**mffs**	FRT	933
63	711	XFL	**mtfsf**	BF, U	936

A.3 Instruction Formats

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions have an extended opcode in another field. Remaining instruction bits contain additional fields. All instruction fields belong to one of the following categories:

- Defined

  These instructions contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

These fields contain operands, such as GPR selectors and immediate values, that can vary from execution to execution. The instruction format diagrams specify the operands in the variable fields.

- Reserved

  Bits in reserved fields should be set to 0. In the instruction format diagrams, /, //, or /// indicate reserved fields.

If any bit in a defined field does not contain the expected value, the instruction is illegal and an illegal instruction exception occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid; its result is architecturally undefined. The PPC440 FPU executes all invalid instruction forms without causing an illegal instruction exception.

A.3.1 Instruction Fields

PPC440 FPU instructions contain various combinations of the following fields, as indicated in the instruction format diagrams that follow the field definitions. Numbers, enclosed in parentheses, that follow the field names indicate bit positions; bit fields are indicated by starting and stopping bit positions separated by colons.

AA (30) Absolute address bit.

- 0 The immediate field represents an address relative to the current instruction address (CIA). The effective address (EA) of the branch is either the sum of the LI field sign-extended to 32 bits and the branch instruction address, or the sum of the BD field sign-extended to 32 bits and the branch instruction address.
- 1 The immediate field represents an absolute address. The EA of the branch is either the LI field or the BD field, sign-extended to 32 bits.

BA (11:15) Specifies a bit in the CR used as a source of a CR-logical instruction.

BB (16:20) Specifies a bit in the CR used as a source of a CR-logical instruction.

BD (16:29) An immediate field specifying a 14-bit signed twos complement branch displacement. This field is concatenated on the right with 0b00 and sign-extended to 32 bits.

BF (6:8) Specifies a field in the CR used as a target in a compare or **mcrf** instruction.

BFA (11:13) Specifies a field in the CR used as a source in a **mcrf** instruction.

BI (11:15) Specifies a bit in the CR used as a source for the condition of a conditional branch instruction.

BO (6:10) Specifies options for conditional branch instructions.

BT (6:10) Specifies a bit in the CR used as a target as the result of a CR-Logical instruction.

D (16:31) Specifies a 16-bit signed twos-complement integer displacement for load/store instructions.

DCRN (11:20) Specifies a device control register (DCR).

FXM (12:19) Field mask used to identify CR fields to be updated by the **mtcrf** instruction.

IM (16:31) An immediate field used to specify a 16-bit value (either signed integer or unsigned).

LI (6:29) An immediate field specifying a 24-bit signed twos complement branch displacement; this field is concatenated on the right with b'00' and sign-extended to 32 bits.

LK (31) Link bit.

- 0 Do not update the link register (LR).
- 1 Update the LR with the address of the next instruction.

MB (21:25) Mask begin.

Used in rotate-and-mask instructions to specify the beginning bit of a mask.

ME (26:30) Mask end.

Used in rotate-and-mask instructions to specify the ending bit of a mask.

NB (16:20) Specifies the number of bytes to move in an immediate string load or store.

OPCD (0:5) Primary opcode. Primary opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The OPCD field name does not appear in instruction descriptions.

OE (21) Enables setting the OV and SO fields in the fixed-point exception register (XER) for extended arithmetic.

RA (11:15) A GPR used as a source or target.

RB (16:20) A GPR used as a source.

Rc (31) Record bit.

0 Do not set the CR.

1 Set the CR to reflect the result of an operation.

RS (6:10) A GPR used as a source.

RT (6:10) A GPR used as a target.

SH (16:20) Specifies a shift amount.

SPRF (11:20) Specifies a special purpose register (SPR).

TO (6:10) Specifies the conditions on which to trap, as described under **tw** and **twi** instructions.

XO (21:30) Extended opcode for instructions without an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

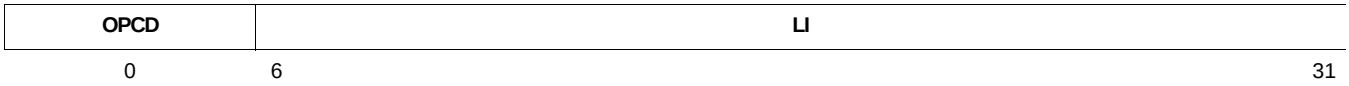
XO (22:30) Extended opcode for instructions with an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

A.3.2 Instruction Format Diagrams

The instruction formats (also called "forms") illustrated in *Figure A-1* through *Figure A-10* are valid combinations of instruction fields. *Table A-2* on page 977 indicates which "form" is utilized by each PPC440 FPU opcode. Fields indicated by slashes (/, //, or ///) are reserved. The figures are adapted from the PowerPC User Instruction Set Architecture.

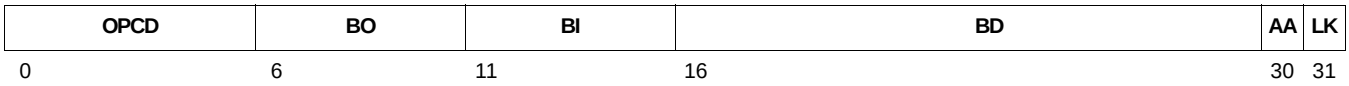
A.3.2.1 I-Form

Figure A-1. I Instruction Format



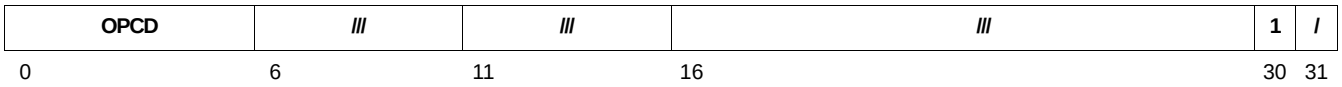
A.3.2.2 B-Form

Figure A-2. B Instruction Format



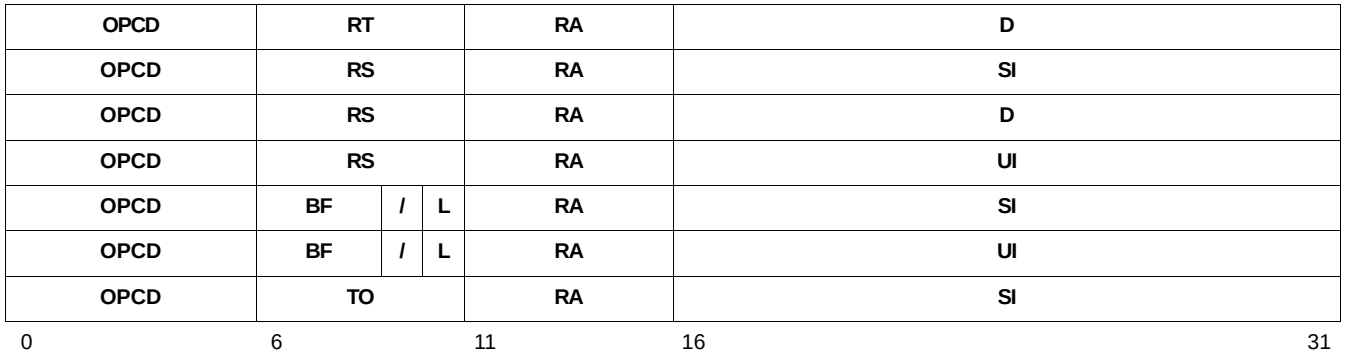
A.3.2.3 SC-Form

Figure A-3. SC Instruction Format



A.3.2.4 D-Form

Figure A-4. D Instruction Format



A.3.2.5 X-Form

Figure A-5. X Instruction Format

OPCD	RT	RA	RB	XO	Rc
OPCD	RT	RA	RB	XO	/
OPCD	RT	RA	NB	XO	/
OPCD	RT	RA	WS	XO	/
OPCD	RT	///	RB	XO	/
OPCD	RT	///	///	XO	/
OPCD	RS	RA	RB	XO	Rc
OPCD	RS	RA	RB	XO	1
OPCD	RS	RA	RB	XO	/
OPCD	RS	RA	NB	XO	/
OPCD	RS	RA	WS	XO	/
OPCD	RS	RA	SH	XO	Rc
OPCD	RS	RA	///	XO	Rc
OPCD	RS	///	RB	XO	/
OPCD	RS	///	///	XO	/
OPCD	BF / L	RA	RB	XO	/
OPCD	BF //	BFA //	///	XO	Rc
OPCD	BF //	///	///	XO	/
OPCD	BF //	///	U	XO	Rc
OPCD	BF //	///	///	XO	/
OPCD	TO	RA	RB	XO	/
OPCD	BT	///	///	XO	Rc
OPCD	///	RA	RB	XO	/
OPCD	///	///	///	XO	/
OPCD	///	///	E //	XO	/

0 6 11 16 21 31

A.3.2.6 XL-Form

Figure A-6. XL Instruction Format

OPCD	BT	BA	BB	XO	/
OPCD	BC	BI	///	XO	LK
OPCD	BF //	BFA //	///	XO	/
OPCD	///	///	///	XO	/

0 6 11 16 21 31

A.3.2.7 XFL-Form

Figure A-7. XFX Instruction Format

OPCD	/	FLM	/	FRB	XO	Rc
0	6	15	16	21		31

A.3.2.8 XFX-Form

Figure A-8. XFX Instruction Format

OPCD	RT	SPRF	XO	/	
OPCD	RT	DCRF	XO	/	
OPCD	RT	/ FXM /	XO	/	
OPCD	RS	SPRF	XO	/	
OPCD	RS	DCRF	XO	/	
0	6	11 16	21	31	

A.3.2.9 X0-Form

Figure A-9. XO Instruction Format

OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	///	/	XO	Rc
0	6	11	16	21	22	31

A.3.2.10 M-Form

Figure A-10. M Instruction Format

OPCD	RS	RA	RB	MB	ME	Rc
OPCD	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

Index

A

addressing 111, 139
addressing modes 140
ANSI/IEEE Standard 754-1985 139

B

bits
  guard (G) 152
  round (R) 152
  sticky (X) 152
boot from NDFC 587
bus timeout error 564

C

CCRYP0_SA_HASH_B 354
chip reset results 169
clock and power management 291
  registers 291
clocking 273
  registers 284
  system 274
    bypass PLL 278
    choosing system clock ratios 279
    clocks for offchip use 277
    CPU feedback example 279, 280
    CPU PLB frequency 278
    feedback selection 275
    IIC bootstrap controller clocking 277
    input system clock 274
    M value for SYS PLL 275
    PerClk feedback example 281
    SYS PLL strapping 278
    SYS PLL TUNE setting 277
    system clock ratio examples 279
    VCO frequency 275
computional instructions 155
condition register 260
conventions
  notational 50
core reset results 169
CPM0_ER 291
CPM0_FR 293
CPM0_SR 294
CPM1_ER 291
CPM1_FR 293
CPM1_SR 294
CPR0 registers 175
CPR0_CFGADDR 284
CPR0_CFGDATA 284
CPR0_CLKUPD 285
CPR0_ICFG 176
CPR0_MALD 289
CPR0_OPBD0 288
CPR0_PERD0 288
CPR0_PLLC0 285
CPR0_PLLD0 286
CPR0_PRIMAD0 287
CPR0_PRIMBD0 288
CPR0_SPCID 289
CRYP0_DC_CTRL 378
CRYP0_DC_DEVID 378
CRYP0_DC_DEVINF 379
CRYP0_DMA_CFG 382
CRYP0_DMA_UCMD 381
CRYP0_DMA_UDST 380
CRYP0_DMA_USRC 380
CRYP0_INT_CFG 376
CRYP0_INT_DESCT 377
CRYP0_INT_DESRD 377
CRYP0_INT_EN 375
CRYP0_INT_MSK 374
CRYP0_INT_UNMSK 374
CRYP0_PE_CTLST 313
CRYP0_PE_DEST 319
CRYP0_PE_DMA_CF 322
CRYP0_PE_DMA_ST 324
CRYP0_PE_E_RING 331
CRYP0_PE_GATH 332
CRYP0_PE_I_RING 330
CRYP0_PE_IO_THR 331
CRYP0_PE_LENGTH 321
CRYP0_PE_PDR_BA 326
CRYP0_PE_PT_CFG 333
CRYP0_PE_PT_DTA 336
CRYP0_PE_PT_DTC 337
CRYP0_PE_PT_S 333
CRYP0_PE_PT_SCA 334
CRYP0_PE_PT_SCC 335
CRYP0_PE_RDR_BA 327
CRYP0_PE_RING_P 329
CRYP0_PE_RING_S 328
CRYP0_PE_SA 320
CRYP0_PE_SCAT 332
CRYP0_PE_SOURCE 319
CRYP0_PKA_COMP 370
CRYP0_PKA_DIV 371
CRYP0_PKA_FUNC 367
CRYP0_PKA_MOD 372
CRYP0_PKA_SHIFT 366
CRYP0_PKA_x_LEN 365
CRYP0_PKA_x_PTR 365
CRYP0_PRNG_CTRL 384
CRYP0_PRNG_KxL/H 385
CRYP0_PRNG_LFL/H 386
CRYP0_PRNG_RSx 386
CRYP0_PRNG_SDL/H 385
CRYP0_PRNG_STAT 383
CRYP0_SA_ARC4IJ 352
CRYP0_SA_ARC4SB 353

CRYP0_SA_CMD_0 338
CRYP0_SA_CMD_1 342
CRYP0_SA_ICV_x 355
CRYP0_SA_IH_Dx 347
CRYP0_SA_IH_Dx (mirror) 354
CRYP0_SA_IV_x 353
CRYP0_SA_KEYx_L/H 346
CRYP0_SA_NONCE 351
CRYP0_SA_OH_Dx 348
CRYP0_SA_PNTR 352
CRYP0_SA_SEQ 350
CRYP0_SA_SEQMKL/H 351
CRYP0_SA_SPI 349
CRYP0_TRNG_ALRM 360
CRYP0_TRNG_CFG 361
CRYP0_TRNG_CNTR 360
CRYP0_TRNG_CTRL 357
CRYP0_TRNG_DATA 356
CRYP0_TRNG_ENTA 358
CRYP0_TRNG_ENTB 359
CRYP0_TRNG_IV_L/H 364
CRYP0_TRNG_K0_L/H 363
CRYP0_TRNG_K1_L/H 364
CRYP0_TRNG_LF0L/H 362
CRYP0_TRNG_LF1L/H 363
CRYP0_TRNG_STAT 356
CRYP0_TRNG_Xx 359

D

data addressing modes 140
data storage addressing modes 140
DCRs 115
  defined 947
DDR SDRAM DCR descriptions 412
DDR_SDRAM controller 405
  address mapping 406
  clocking 434
  delay compensation circuit 439
  error checking and correction 435
  initialization 405
  read data capture 440
  registers 410
  write data timing 444
DDR0_00 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434
DDR0_01 413
DDR0_02 414
DDR0_03 415
DDR0_04 416
DDR0_05 416
DDR0_06 417
DDR0_07 418
DDR0_08 419
DDR0_09 420
DDR0_10 421
DDR0_11 422
DDR0_12 422
DDR0_14 423
DDR0_17 424
DDR0_18 424
DDR0_19 425
DDR0_20 425
DDR0_21 426
DDR0_22 427
DDR0_23 428
DDR0_24 429
DDR0_25 429
DDR0_26 430
DDR0_27 430
DDR0_28 430
DDR0_32 431
DDR0_33 431
DDR0_34 431
DDR0_35 431
DDR0_36 431
DDR0_37 431
DDR0_38 432
DDR0_39 432
DDR0_40 432
DDR0_41 432
DDR0_42 433
DDR0_43 433
DDR0_44 434
denormalization 149
device control registers 115, 947
Device Control Registers. *See also* DCRs
device control registers. See DCRs 115
device-paced transfers 563
  bus timeout error 564
DMA
  memory-to-memory mode transfers
    initiated by software 644
DMA controller 609
  configuration and status registers 611
  transfers 609
DMA operations
  arbitration transfer priorities 636
  channel priorities 619
  data parity 619, 637
  errors 620, 637
  interrupts 621, 638
  peripheral and deviced paced memory bursts 619
  programming 623
  scatter gather transfers 622
DMA2P30_ADR 636
DMA2P30_CRx 632
DMA2P30_CTx 634
DMA2P30_DAx 634
DMA2P30_SAx 634
DMA2P30_SCx 636
DMA2P30_SGC 635
DMA2P30_SGx 635
DMA2P30_SLP 631
DMA2P30_SR 631
DMA2P40_CRx 613

DMA2P40_CTx 614
DMA2P40_DAHx 616
DMA2P40_DALx 616
DMA2P40_POL 619
DMA2P40_SAHx 615
DMA2P40_SALx 615
DMA2P40_SGC 618
DMA2P40_SGHx 616
DMA2P40_SGLx 616
DMA2P40_SLP 618
DMA2P40_SR 617

E

EBC
  signals 555
EBC error reporting 580
EBC registers 574
EBC0_BEAR 580
EBC0_BESR0 580
EBC0_BxAP 578
EBC0_BxCR 577
EBC0_CFG 576
ECC 592, 600
EHCI0_ASYNCLSTA 830
EHCI0_CFGFLAG 830
EHCI0_CRTLSEG 829
EHCI0_FRINDEX 828
EHCI0_HCCAPBASE 820
EHCI0_HCCPARAMS 822
EHCI0_HCSPARAMS 820
EHCI0_INSNREG0 835
EHCI0_INSNREG1 836
EHCI0_INSNREG2 837
EHCI0_INSNREG3 837
EHCI0_INSNREG4 838
EHCI0_PORTSC 831
EHCI0_PRDLISTB 829
EHCI0_USBCMD 823
EHCI0_USBINTR 827
EHCI0_USBSTS 825
EMAC to PHY bridge 681
EMACx_GAHT1-GAHT4 734
EMACx_IAHR 732
EMACx_IAHT1-IAHT4 734
EMACx_IALR 732
EMACx_IPGVR 735
EMACx_ISER 730
EMACx_ISR 728
EMACx_LSAH 734
EMACx_LSAL 734
EMACx_MR0 723
EMACx_MR1 723
EMACx_OCRX 738
EMACx_OCTX 738
EMACx_PTR 733
EMACx_RMR 726
EMACx_RWMR 737
EMACx_STACR 735
EMACx_TMR0 725
EMACx_TMR1 726
EMACx_TRTR 736
EMACx_VTCI 733
EMACx_VTPID 733
error correction code (implemented in NDFC) 592, 600
error handling, PCI interface 493
error reporting, EBC 580
Ethernet MAC 693
  features 695
  flow control 712
  MAL- MAC packet transfer flow 739
  MII 738
  operations 696
  receive operation 709
  registers 720
  transmit operation 703
  VLAN support 715
exception
  floating-point 141
  inexact 141, 159, 250, 257
  invalid operation 253
  overflow 141, 159, 250, 255
  underflow 141, 159, 250, 256
  updating the CR 260
  updating the FPR 259
  zero divide 141, 250, 255
exception priorities for floating-point load and store instructions 258
exception syndrome register (ESR) 249
exceptions 247
execution model
  multply-add 64-bit 154
execution model for multiply-add type instructions 154
external bus controller
  signals 555
external bus master 568
  arbitration 569
  interface 568
extranl bus controller 555

F

fabs 895
fadd 896
fadds 897
fcmpo 898
fctiw 900
fctiwz 901
fdiv 902
fdivs 903
FEX 143
field
  n 260
Flash controller, NAND 583
floating point registers 142
floating-point

denormalization 149
denormalized number 147
infinity 148
normalization 149
Not a Number 148
sign 149
zero 147
floating-point accumulator 154
floating-point compare and select instruction set index 161
floating-point compare instructions
comparison sets 160
floating-point instructions 894
floating-point multiply-add instructions 159
floating-point operands 156
double precision format 156
single format 156
floating-point programming model 139
floating-point rounding and conversion instruction set index 160
floating-point status and control register 161
instruction set index 161
floating-point unavailable interrupt 252
fmadd 904
fmadds 905
fmr 906
fmsub 907
fmsubs 908
fmsubs. 908
fmul 909
fmuls 910
fnabs 911
fneg 912
fnmadd 913
fnmadds 914
fnmsub 915
fnmsubs 916
fpcmpu 899
FPR0:FPR31 142
FPSCR 142
fres 917
fres. 917
frsp 918
frsqrte 919
fsel 921
fsub 922
fsubs 923

G

general purpose timers 265
registers 266
GMII 738
GPIO0_IRO 795
GPT0_COMP0:GPT0_COMP6 270
GPT0_DCIS 271
GPT0_DCT0 270
GPT0_IE 269
GPT0_IM 268
GPT0_ISS, GPT0_ISC 268
GPT0_MASK0:GPT0_MASK6 270
GPT0_TBC 267

I

IIC bootstrap controller 187
configuration 187
IIC bus interface 775
addressing modes 775
interrupt handling 792
registers 777
IICx_CLKDIV 786
IICx_CNTL 780
IICx_DIRECTCNTL 790
IICx_EXTSTS 784
IICx_HMADR 780
IICx_HSADR 786
IICx_INTRMSK 787
IICx_LMADR 779
IICx_LSADR 785
IICx_MDBUF 778
IICx_MDCNTL 781
IICx_SDBUF 778
IICx_STS 782
IICx_XFRCNT 788
IICx_XTCNTLSS 789
initialization 175
instruction fields 979
instruction formats 891, 978
diagrams 980
instruction forms 978, 980
instruction set portability 891
instructions
alphabetical, including extended mnemonics 973
by category 155
categories 891
computational 155
fabs 895
fadd 896
fadds 897
fcmpo 898
fcmpu 899
fctiw 900
fctiwz 901
fdiv 902
fdivs 903
fmadd 904
fmadds 905
fmr 906
fmsub 907
fmsubs 908
fmsubs. 908
fmul 909
fmuls 910
fnabs 911
fneg 912
fnmadd 913

fnmadds 914
fnmsub 915
fnmsubs 916
format diagrams 980
formats 978
forms 978, 980
fres 917
fres. 917
frsp 918
frsqrte 919
fsel 921
fsub 922
fsubs 923
lfd 924
lfdu 925
lfdux 926
lfdx 927
lfs 928
lfsu 929
lfsux 930
lfsx 931
mcrfs 932
mffs 933
mtfsb0 934
mtfsb0. 934
mtfsb1 935
mtsfs 936
mtsfsi 937
noncomputational 155
opcodes 977
stfd 938
stfdu 939
stfdux 940
stfdx 941
stfiwx 942
stfs 943
stfsu 944
stfsx 946
interrupt
type
floating-point unavailable 252
interrupts 247
invalid operation exception (SNaN) 250
invalid operation exception bit 160

K

KASU0_CONFIG 398
KASU0_COUNT 398
KASU0_CTRL 395
KASU0_DATAIN0 394
KASU0_DATAIN1 394
KASU0_DATAOUT0 394
KASU0_DATAOUT1 394
KASU0_FRESH 399
KASU0_KEYx 397
KASU0_MODE 396
KASU0_STAT 395
kasumi algorithm 389
block diagram 389
functional description 390
programming examples 399
register interface 393

L

lfd 924
lfdu 925
lfdux 926
lfdx 927
lfs 928
lfsu 929
lfsux 930
lfsx 931

M

MAL0_CFG 669
MAL0_ESR 672
MAL0_IER 674
MAL0_RCBSx 677
MAL0_RXCARR 670
MAL0_RXCASR 670
MAL0_RXCTPxR 676
MAL0_RXDEIR 674, 676
MAL0_RXEOBISR 671
MAL0_RXTATTRR 675
MAL0_TXCARR 670
MAL0_TXCASR 670
MAL0_TXCTPxR 676
MAL0_TXDEIR 674, 676
MAL0_TXEOBISR 671
MAL0_TXTATTRR 675
mcrfs 932
media access layer
programming 662
registers 668
memory access layer 645
buffer descriptor 652
descriptor buffer status fields, descriptor buffer control fields 658
features 645
interfaces and channel assignments 648
receive software interface 657
transmit and receive operations 649
transmit software interface 654
memory interface
bus attachment
alternative 556
external bus master 568
SRAM
burst mode 560
bus timeout error 564
device-paced transfers 563
memory management. *See also* MMU

memory map 111, 139
memory map, NDFC register 600
memory mapped registers 125
memory organization 111, 139
mffs 933
MII 738
mtfsb0 934
mtfsb0. 934
mtfsb1 935
mtsfs 936
mtsfsi 937

N

NAND Flash controller 583
NDFC interface
  overview 584
NDFC registers 600
NDFC, boot from 587
NDFC0_ADDR 601
NDFC0_B0CR:NDFC0_B3CR 603
NDFC0_CMD 601
NDFC0_CR 604
NDFC0_DATA 601
NDFC0_ECC0:NDFC0_ECC7 603
NDFC0_HWCTL 607
NDFC0_REVID 608
NDFC0_SR 606
noncomputional instructions 155
notation 50, 892, 979
notational conventions 50

O

OHCI0_HCBULKCED 847
OHCI0_HCBULKHED 847
OHCI0_HCCHED 846
OHCI0_HCCMDSTS 840
OHCI0_HCCTRL 839
OHCI0_HCCTRLCED 846
OHCI0_HCDHEAD 848
OHCI0_HCFMINT 849
OHCI0_HCFMNUM 850
OHCI0_HCFMREM 850
OHCI0_HCHCCA 845
OHCI0_HCINTDIS 844
OHCI0_HCINTE 843
OHCI0_HCINTSTS 842
OHCI0_HCLSTHRES 851
OHCI0_HCPCED 845
OHCI0_HCPRDSTRT 851
OHCI0_HCPRSTS 856
OHCI0_HCREV 838
OHCI0_HCRHDESCA 852
OHCI0_HCRHDESCB 854
OHCI0_HCRHSTS 855
on-chip buses 57
  DCR bus 107
  OPB arbiter registers 103
  OPB bus 102
  OPB master assignments 103
  OPB to PLB4 registers 105
  PLB4 arbiter registers 69
  PLB4 to OPB Bridge registers 90
  processor local bus 57
    master and slave assignments 58
    master priority assignments 59
OPB2PLB40_BCTRL 105
OPB2PLB40_BEARH 106
OPB2PLB40_BEARL 106
OPB2PLB40_BSTAT 105
OPB2PLB40_REVID 106
OPBAx_CR 104
OPBAx_PR 103
opcodes 977

P

P3P4BI0_BEARH 85
P3P4BI0_BEARL 85
P3P4BI0_BESR0 85
P3P4BI0_BESR1 87
P3P4BI0_CFG 88
P3P4BI0_PEIR 90
P3P4BI0_PICR 89
P3P4BI0_REVID 90
P4P3BO0_BEARH 77
P4P3BO0_BEARL 77
P4P3BO0_BESR0 77
P4P3BO0_BESR1 80
P4P3BO0_CFG 82
P4P3BO0_PEIR 84
P4P3BO0_PICR 83
P4P3BO0_REVID 84
PCI bridge completion ordering rules 469
PCI bridge configuration registers 471
PCI bridge register summary 471
PCI bridge reset and initialization 498
PCI bridge timing diagrams 501
PCI configuration registers
  offsets 472
PCI interface 455
  arbiter 458
  block diagram 456
  bridge address mapping 459
  bridge functional blocks 457
  bridge transaction handling 462
  byte ordering 456
  error handling 493
  master commands 463
  target map configuration 462
PCI power management interface 497
PCIC0_BIST 482
PCIC0_BRDGOPT1 487
PCIC0_BRDGOPT2 492

PCIC0_CACHELS 482
PCIC0_CAP 484
PCIC0_CAPID 490
PCIC0_CFGADDR 478
PCIC0_CFGDATA 478
PCIC0_CLS 481
PCIC0_CMD 479
PCIC0_DATA 492
PCIC0_DEVID 479
PCIC0_ERREN 485
PCIC0_ERRSTS 486
PCIC0_HDTYPE 482
PCIC0_ICS 485
PCIC0_INTLN 484
PCIC0_INTPN 485
PCIC0_LATTIM 482
PCIC0_MAXLTNCY 485
PCIC0_MINGNT 485
PCIC0_NEXTIPTR 490
PCIC0_PLBBEAR 490
PCIC0_PLBBESR0 487
PCIC0_PLBBESR1 489
PCIC0_PMC 491
PCIC0_PMCSR 491
PCIC0_PMCSRBSE 492
PCIC0_PMSCRR 493
PCIC0_PTM1BAR 483
PCIC0_PTM2BAR 483
PCIC0_REVID 481
PCIC0_SBSYSID 484
PCIC0_SBSYSVID 484
PCIC0_STATUS 480
PCIC0_VENDID 479
PCIL0_PMM0LA 473
PCIL0_PMM0MA 473
PCIL0_PMM0PCIHA 474
PCIL0_PMM0PCILA 474
PCIL0_PMM1LA 474
PCIL0_PMM1MA 474
PCIL0_PMM1PCIHA 475
PCIL0_PMM1PCILA 475
PCIL0_PMM2LA 475
PCIL0_PMM2MA 475
PCIL0_PMM2PCIHA 476
PCIL0_PMM2PCILA 476
PCIL0_PTM1LA 476
PCIL0_PTM1MS 476
PCIL0_PTM2LA 477
PCIL0_PTM2MS 477
PCI-to-PLB half-bridge 458
PCI-to-PLB transaction handling 466
peripheral component interconnect interface 455
PLB master commands 467
PLB3 arbiter
  registers 96
PLB3 to OPB bridge
  registers 99
PLB3 to OPB bridge error status register 0 100
PLB3 to OPB bridge error status register 1 102
PLB3 to PLB4 bridge
  registers 84
PLB32OPB0_BEAR 101
PLB32OPB0_BESR0 100
PLB32OPB0_BESR1 102
PLB3A0_ACR 98
PLB3A0_BEAR 98
PLB3A0_BESR 97
PLB3A0_REVID 96
PLB4 to PLB3 bridge
  registers 76
PLB42OPBx_BEARH 93
PLB42OPBx_BEARL 93
PLB42OPBx_BESR0 91
PLB42OPBx_BESR1 94
PLB42OPBx_CFG 95
PLB42OPBx_LATENCY 95
PLB42OPBx_REVID 96
PLB4A0_CCR 75
PLB4A0_REVID 69
PLB4An_ACR 70
PLB4An_EARH 75
PLB4An_EARL 75
PLB4An_ESRL 73
PLB-to-PCI half-bridge 457
portability, instruction set 891
PPC440GP
  signals 971
primary opcodes 977
pseudocode 892

Q

QNaN 259

R

registers 141
  clock and power management 291
  clocking 284
  CPM0_ER 291
  CPM0_FR 293
  CPM0_SR 294
  CPM1_ER 291
  CPM1_FR 293
  CPM1_SR 294
  CPR0_CFGADDR 284
  CPR0_CFGDATA 284
  CPR0_CLKUPD 285
  CPR0_ICFG 176
  CPR0_MALD 289
  CPR0_OPBD0 288
  CPR0_PERD0 288
  CPR0_PLLC0 285
  CPR0_PLLD0 286
  CPR0_PRIMAD0 287
  CPR0_PRIMBD0 288

CPR0_SPCID 289
CRYP0_DC_CTRL 378
CRYP0_DC_DEVID 378
CRYP0_DC_DEVINF 379
CRYP0_DMA_CFG 382
CRYP0_DMA_UCMD 381
CRYP0_DMA_UDST 380
CRYP0_DMA_USRC 380
CRYP0_INT_CFG 376
CRYP0_INT_DESCT 377
CRYP0_INT_DESRD 377
CRYP0_INT_EN 375
CRYP0_INT_MSK 374
CRYP0_INT_UNMSK 374
CRYP0_PE_CTLST 313
CRYP0_PE_DEST 319
CRYP0_PE_DMA_CF 322
CRYP0_PE_DMA_ST 324
CRYP0_PE_E_RING 331
CRYP0_PE_GATH 332
CRYP0_PE_I_RING 330
CRYP0_PE_IO_THR 331
CRYP0_PE_LENGTH 321
CRYP0_PE_PDR_BA 326
CRYP0_PE_PT_CFG 333
CRYP0_PE_PT_DTA 336
CRYP0_PE_PT_DTC 337
CRYP0_PE_PT_S 333
CRYP0_PE_PT_SCA 334
CRYP0_PE_PT_SCC 335
CRYP0_PE_RDR_BA 327
CRYP0_PE_RING_P 329
CRYP0_PE_RING_S 328
CRYP0_PE_SA 320
CRYP0_PE_SCAT 332
CRYP0_PE_SOURCE 319
CRYP0_PKA_COMP 370
CRYP0_PKA_DIV 371
CRYP0_PKA_FUNC 367
CRYP0_PKA_MOD 372
CRYP0_PKA_SHIFT 366
CRYP0_PKA_x_LEN 365
CRYP0_PKA_x_PTR 365
CRYP0_PRNG_CTRL 384
CRYP0_PRNG_KxL/H 385
CRYP0_PRNG_LFL/H 386
CRYP0_PRNG_RSx 386
CRYP0_PRNG_SDL/H 385
CRYP0_PRNG_STAT 383
CRYP0_SA_ARC4IJ 352
CRYP0_SA_ARC4SB 353
CRYP0_SA_CMD_0 338
CRYP0_SA_CMD_1 342
CRYP0_SA_HASH_B 354
CRYP0_SA_ICV_x 355
CRYP0_SA_IH_Dx 347
CRYP0_SA_IH_Dx (mirror) 354
CRYP0_SA_IV_x 353
CRYP0_SA_KEYx_L/H 346
CRYP0_SA_NONCE 351
CRYP0_SA_OH_Dx 348
CRYP0_SA_PNTR 352
CRYP0_SA_SEQ 350
CRYP0_SA_SEQMKL/H 351
CRYP0_SA_SPI 349
CRYP0_TRNG_ALRM 360
CRYP0_TRNG_CFG 361
CRYP0_TRNG_CNTR 360
CRYP0_TRNG_CTRL 357
CRYP0_TRNG_DATA 356
CRYP0_TRNG_ENTA 358
CRYP0_TRNG_ENTB 359
CRYP0_TRNG_IV_L/H 364
CRYP0_TRNG_K0_L/H 363
CRYP0_TRNG_K1_L/H 364
CRYP0_TRNG_LF0L/H 362
CRYP0_TRNG_LF1L/H 363
CRYP0_TRNG_STAT 356
CRYP0_TRNG_Xx 359
DCR 115
DDR0_00 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434
DDR0_01 413
DDR0_02 414
DDR0_03 415
DDR0_04 416
DDR0_05 416
DDR0_06 417
DDR0_07 418
DDR0_08 419
DDR0_09 420
DDR0_10 421
DDR0_11 422
DDR0_12 422
DDR0_14 423
DDR0_17 424
DDR0_18 424
DDR0_19 425
DDR0_20 425
DDR0_21 426
DDR0_22 427
DDR0_23 428
DDR0_24 429
DDR0_25 429
DDR0_26 430
DDR0_27 430
DDR0_28 430
DDR0_32 431
DDR0_33 431
DDR0_34 431
DDR0_35 431
DDR0_36 431
DDR0_37 431
DDR0_38 432
DDR0_39 432
DDR0_40 432
DDR0_41 432

DDR0_42 433
DDR0_43 433
DDR0_44 434
DMA2P30_ADR 636
DMA2P30_CRx 632
DMA2P30_CTx 634
DMA2P30_DAx 634
DMA2P30_POL 630
DMA2P30_SAx 634
DMA2P30_SCx 636
DMA2P30_SGC 635
DMA2P30_SGx 635
DMA2P30_SLP 631
DMA2P30_SR 631
DMA2P40_CRx 613
DMA2P40_CTx 614
DMA2P40_DAHx 616
DMA2P40_DALx 616
DMA2P40_POL 619
DMA2P40_SAHx 615
DMA2P40_SALx 615
DMA2P40_SGC 618
DMA2P40_SGHx 616
DMA2P40_SGLx 616
DMA2P40_SLP 618
DMA2P40_SR 617
EBC 574
EBC0_BEAR 580
EBC0_BESR0 580
EBC0_BxAP 578
EBC0_BxCR 577
EBC0_CFG 576
EBC0_CFGADDR 574
EBC0_CFGDATA 574
EHCI0_ASYNCLSTA 830
EHCI0_CFGFLAG 830
EHCI0_CRTLSEG 829
EHCI0_FRINDEX 828
EHCI0_HCCAPBASE 820
EHCI0_HCCPARAMS 822
EHCI0_HCSPARAMS 820
EHCI0_INSNREG0 835
EHCI0_INSNREG1 836
EHCI0_INSNREG2 837
EHCI0_INSNREG3 837
EHCI0_INSNREG4 838
EHCI0_PORTSC 831
EHCI0_PRDLISTB 829
EHCI0_USBCMD 823
EHCI0_USBINTR 827
EHCI0_USBSTS 825
EMACx_GAHT1-GAHT4 734
EMACx_IAHR 732
EMACx_IAHT1-IAHT4 734
EMACx_IALR 732
EMACx_IPGVR 735
EMACx_ISER 730
EMACx_ISR 728
EMACx_LSAH 734
EMACx_LSAL 734
EMACx_MR0 723
EMACx_MR1 723
EMACx_OCRX 738
EMACx_OCTX 738
EMACx_PTR 733
EMACx_RMR 726
EMACx_RWMR 737
EMACx_STACR 735
EMACx_TMR0 725
EMACx_TMR1 726
EMACx_TRTR 736
EMACx_VTCI 733
EMACx_VTPID 733
Ethernet MAC 720
FPR0:FPR31 142
FPSCR 142
GPT0_COMP0:GPT0_COMP6 270
GPT0_DCIS 271
GPT0_DCT0 270
GPT0_IE 269
GPT0_IM 268
GPT0_ISS, GPT0_ISC 268
GPT0_MASK0:GPT0_MASK6 270
GPT0_TBC 267
IIC bus interface 777
IICx_CLKDIV 786
IICx_CNTL 780
IICx_DIRECTCNTL 790
IICx_EXTSTS 784
IICx_HMADR 780
IICx_HSADR 786
IICx_INTRMSK 787
IICx_LMADR 779
IICx_LSADR 785
IICx_MDBUF 778
IICx_MDCNTL 781
IICx_SDBUF 778
IICx_STS 782
IICx_XFRCNT 788
IICx_XTCNTLSS 789
KASU0_CONFIG 398
KASU0_COUNT 398
KASU0_CTRL 395
KASU0_DATAIN0 394
KASU0_DATAIN1 394
KASU0_DATAOUT0 394
KASU0_DATAOUT1 394
KASU0_FRESH 399
KASU0_KEYx 397
KASU0_MODE 396
KASU0_STAT 395
MAL0_CFG 669
MAL0_ESR 672
MAL0_IER 674
MAL0_RCBSx 677
MAL0_RXCARR 670
MAL0_RXCASR 670
MAL0_RXCTPxR 676

MAL0_RXDEIR 674, 676
MAL0_RXEOBISR 671
MAL0_RXTATTRR 675
MAL0_TXCARR 670
MAL0_TXCASR 670
MAL0_TXCTPxR 676
MAL0_TXDEIR 674, 676
MAL0_TXEOBISR 671
MAL0_TXTATTRR 675
media access layer 668
NDFC0_ADDR 601
NDFC0_B0CR:NDFC0_B3CR 603
NDFC0_CMD 601
NDFC0_CR 604
NDFC0_DATA 601
NDFC0_ECC0:NDFC0_ECC7 603
NDFC0_HWCTL 607
NDFC0_REVID 608
NDFC0_SR 606
OHCI0_HCBULKCED 847
OHCI0_HCBULKHED 847
OHCI0_HCCHED 846
OHCI0_HCCMDSTS 840
OHCI0_HCCTRL 839
OHCI0_HCCTRLCED 846
OHCI0_HCDHEAD 848
OHCI0_HCFMINT 849
OHCI0_HCFMNUM 850
OHCI0_HCFMREM 850
OHCI0_HCHCCA 845
OHCI0_HCINTDIS 844
OHCI0_HCINTE 843
OHCI0_HCINTSTS 842
OHCI0_HCLSTHRES 851
OHCI0_HCPCED 845
OHCI0_HCPRDSTRT 851
OHCI0_HCPRSTS 856
OHCI0_HCREV 838
OHCI0_HCRHDESCA 852
OHCI0_HCRHDESCB 854
OHCI0_HCRHSTS 855
OPB2PLB40_BCTRL 105
OPB2PLB40_BEARH 106
OPB2PLB40_BEARL 106
OPB2PLB40_BSTAT 105
OPB2PLB40_REVID 106
OPBAx_CR 104
OPBAx_PR 103
P3P4BI0_BEARH 85
P3P4BI0_BEARL 85
P3P4BI0_BESR0 85
P3P4BI0_BESR1 87
P3P4BI0_CFG 88
P3P4BI0_PEIR 90
P3P4BI0_PICR 89
P3P4BI0_REVID 90
P4P3BO0_BEARH 77
P4P3BO0_BEARL 77
P4P3BO0_BESR0 77
P4P3BO0_BESR1 80
P4P3BO0_CFG 82
P4P3BO0_PEIR 84
P4P3BO0_PICR 83
P4P3BO0_REVID 84
PCIC0_BIST 482
PCIC0_BRDGOPT1 487
PCIC0_BRDGOPT2 492
PCIC0_CACHELS 482
PCIC0_CAP 484
PCIC0_CAPID 490
PCIC0_CFGADDR 478
PCIC0_CFGDATA 478
PCIC0_CLS 481
PCIC0_CMD 479
PCIC0_DATA 492
PCIC0_DEVID 479
PCIC0_ERREN 485
PCIC0_ERRSTS 486
PCIC0_HDTYPE 482
PCIC0_ICS 485
PCIC0_INTLN 484
PCIC0_INTPN 485
PCIC0_LATTIM 482
PCIC0_MAXLTNCY 485
PCIC0_MINGNT 485
PCIC0_NEXTIPTR 490
PCIC0_PLBBEAR 490
PCIC0_PLBBESR0 487
PCIC0_PLBBESR1 489
PCIC0_PMC 491
PCIC0_PMCSR 491
PCIC0_PMCSRBSE 492
PCIC0_PMSCRR 493
PCIC0_PTM1BAR 483
PCIC0_PTM2BAR 483
PCIC0_REVID 481
PCIC0_SBSYSID 484
PCIC0_SBSYSVID 484
PCIC0_STATUS 480
PCIC0_VENDID 479
PCIL0_PMM0LA 473
PCIL0_PMM0MA 473
PCIL0_PMM0PCIHA 474
PCIL0_PMM0PCILA 474
PCIL0_PMM1LA 474
PCIL0_PMM1MA 474
PCIL0_PMM1PCIHA 475
PCIL0_PMM1PCILA 475
PCIL0_PMM2LA 475
PCIL0_PMM2MA 475
PCIL0_PMM2PCIHA 476
PCIL0_PMM2PCILA 476
PCIL0_PTM1LA 476
PCIL0_PTM1MS 476
PCIL0_PTM2LA 477
PCIL0_PTM2MS 477
PLB3 arbiter 96
PLB3 to OPB bridge 99

PLB3 to PLB4 bridge 84
PLB32OPB0_BEAR 101
PLB32OPB0_BESR0 100
PLB32OPB0_BESR1 102
PLB3A0_ACR 98
PLB3A0_BEAR 98
PLB3A0_BESR 97
PLB3A0_REVID 96
PLB4 to PLB3 bridge 76
PLB42OPBx_BEARH 93
PLB42OPBx_BEARL 93
PLB42OPBx_BESR0 91
PLB42OPBx_BESR1 94
PLB42OPBx_CFG 95
PLB42OPBx_LATENCY 95
PLB42OPBx_REVID 96
PLB4A0_CCR 75
PLB4A0_REVID 69
PLB4An_ACR 70
PLB4An_EARH 75
PLB4An_EARL 75
PLB4An_ESRL 73
RGMII0_FER 690
RGMII0_SSR 690
SDR0_AMP0 60
SDR0_AMP1 62
SDR0_CP440 63
SDR0_CRYP0 312
SDR0_CUST0 203
SDR0_CUST1 204
SDR0_DDRCFG 182
SDR0_EBC0 204
SDR0_ECID0 177
SDR0_ECID1 177
SDR0_ECID2 177
SDR0_ECID3 177
SDR0_EMACxREJCNT 702
SDR0_EMACxRXST 699
SDR0_EMACxTXST 701
SDR0_ICCRRX 679
SDR0_ICCRTX 678
SDR0_ICSRRXx 680
SDR0_ICSRTXx 680
SDR0_ICTRRXx 680
SDR0_ICTRTXx 680
SDR0_MALRBL 651
SDR0_MALRBS 651
SDR0_MALTBL 652
SDR0_MALTBS 652
SDR0_MFR 181
SDR0_MIRQ0 64
SDR0_MIRQ1 64
SDR0_PCI0 470
SDR0_PFC0 177
SDR0_PFC1 178
SDR0_PFC2 691
SDR0_PFC4 179
SDR0_PINSTP 197
SDR0_SDCS0 197
SDR0_SDSTP0 198
SDR0_SDSTP1 200
SDR0_SDSTP2 201
SDR0_SDSTP3 202
SDR0_SLPIPE0 67, 179
SDR0_SRSTx 179
SDR0_UART0 754
SDR0_UART1 755
SDR0_UART2 756
SDR0_UART3 756
SDR0_USB2D0CR 812
SDR0_USB2H0CR 811
SDR0_USB2H0ST 811
SDR0_USB2PHY0CR 810
serial peripheral interface 771
SPI0_CDM 773
SPI0_CR 772
SPI0_MODE 774
SPI0_RxD 772
SPI0_SR 773
SPI0_TxD 772
SRAM0_BEAR 448
SRAM0_BESR0 449
SRAM0_BESR1 450
SRAM0_CID 452
SRAM0_DPC 453
SRAM0_PMEG 452
SRAM0_REVID 453
SRAM0_SB0CR 448
types 142
  DCR 947
UART controller 746
UARTx_DLL 753
UARTx_DLM 753
UARTx_FCR 749
UARTx_IER 747
UARTx_IIR 748
UARTx_LCR 749
UARTx_LSR 751
UARTx_MCR 750
UARTx_MSR 752
UARTx_RBR 747
UARTx_SCR 753
UARTx_THR 747
UIC controller 210
UIC0_CR 221
UIC0_ER 215
UIC0_MSR 237
UIC0_PR 226
UIC0_SR 210
UIC0_TR 232
UIC0_VCR 243
UIC0_VR 244
UIC1_CR 223
UIC1_ER 218
UIC1_MSR 240
UIC1_PR 229, 231
UIC1_SR 212
UIC1_TR 234

UIC1_VCR 243
UIC1_VR 245
UIC2_CR 225
UIC2_ER 220
UIC2_MSR 242
UIC2_SR 214
UIC2_TR 237
USB2D0_FADDR 869
USB2D0_FRAME 875
USB2D0_INCSR 878
USB2D0_INCSR0 877
USB2D0_INDEX 875
USB2D0_INMAXP 880
USB2D0_INTRIN 867
USB2D0_INTRINE 870
USB2D0_INTROUT 870
USB2D0_INTROUTE 873
USB2D0_INTRUSB 872
USB2D0_INTRUSBE 871
USB2D0_OUTCOUNT 884
USB2D0_OUTCOUNT0 884
USB2D0_OUTCSR 881
USB2D0_OUTMAXP 883
USB2D0_POWER 868
USB2D0_TSTMODE 874
USB2HMDCR0_BCNTL 858
USB2HMDCR0_BSTAT 859
USB2HMDCR0_LAT 860
USB2HMDCR0_REVID 861
USB2HSDCR0_BCNTL 862
USB2HSDCR0_REVID 863
ZMII bridge 684, 689
ZMII0_FER 685
ZMII0_SMIISR 686
ZMII0_SSR 685
registers, device control 115, 947
registers, memory mapped 125
registers, summary 141
reset types
chip 169
core 169
system 170
resets
signals 169
types 169
RGMII0_FER 690
RGMII0_SSR 690

S

SDR0 registers 175
SDR0_AMP0 60
SDR0_AMP1 62
SDR0_CP440 63
SDR0_CRYP0 312
SDR0_CUST0 203
SDR0_CUST1 204
SDR0_DDRCFG 182
SDR0_EBC0 204
SDR0_ECID0 177
SDR0_ECID2 177
SDR0_ECID3 177
SDR0_EMACxREJCNT 702
SDR0_EMACxRXST 699
SDR0_EMACxTXST 701
SDR0_ICCRRX 679
SDR0_ICCRTX 678
SDR0_ICSRRXx 680
SDR0_ICSRTXx 680
SDR0_ICTRRXx 680
SDR0_ICTRTXx 680
SDR0_MALRBL 651
SDR0_MALRBS 651
SDR0_MALTBL 652
SDR0_MALTBS 652
SDR0_MFR 181
SDR0_MIRQ0 64
SDR0_MIRQ1 64
SDR0_PCI0 470
SDR0_PFC0 177
SDR0_PFC1 178
SDR0_PFC2 691
SDR0_PFC4 179
SDR0_PINSTP 197
SDR0_SCID1 177
SDR0_SDCS0 197
SDR0_SDSTP0 198
SDR0_SDSTP1 200
SDR0_SDSTP2 201
SDR0_SDSTP3 202
SDR0_SLPIPE0 67, 179
SDR0_SRSTx 179
SDR0_UART0 754
SDR0_UART1 755
SDR0_UART2 756
SDR0_UART3 756
SDR0_USB2D0CR 812
SDR0_USB2H0CR 811
SDR0_USB2H0ST 811
SDR0_USB2PHY0CR 810
secondary opcodes 977
security 299
block diagram 299
functional description 299
register interface 306
serial peripheral interface 769
exchange of data 771
interrupt operation 770
loopback operation 770
registers 771
serial port operations
UART controller 743
signals 971
EBC 555
resets 169
SPI 769
SPI0_CDM 773

SPI0_CR 772
SPI0_MODE 774
SPI0_RxD 772
SPI0_SR 773
SPI0_TxD 772
SRAM controller 447
  errors 453
  registers 447
SRAM0_BEAR 448
SRAM0_BESR0 449
SRAM0_BESR1 450
SRAM0_CID 452
SRAM0_DPC 453
SRAM0_PMEG 452
SRAM0_REVID 453
SRAM0_SB0CR 448
stfd 938
stfdu 939
stfdux 940
stfdx 941
stfiwx 942
stfs 943
stfsu 944
stfsx 946
system reset results 170

T

timing diagrams, PCI bridge 501

U

UART controller
  clocking 744
  DMA operation 759
  features 743
  FIFO 757
  registers 746
  sleep mode 758
UART receiver DMA mode 764
UART transmitter DMA mode 760
UARTx_DLL 753
UARTx_DLM 753
UARTx_FCR 749
UARTx_IER 747
UARTx_IIR 748
UARTx_LCR 749
UARTx_LSR 751
UARTx_MCR 750
UARTx_MSR 752
UARTx_RBR 747
UARTx_SCR 753
UARTx_THR 747
UIC controller 205
  features 205
  interrupt assignments 206
  programming 209
  registers 210
UIC0_CR 221
UIC0_ER 215
UIC0_MSR 237
UIC0_PR 226
UIC0_SR 210
UIC0_TR 232
UIC0_VCR 243
UIC0_VR 244
UIC1_CR 223
UIC1_ER 218
UIC1_MSR 240
UIC1_PR 229
UIC1_SR 212
UIC1_TR 234
UIC1_VCR 243
UIC1_VR 245
UIC2_CR 225
UIC2_ER 220
UIC2_MSR 242
UIC2_PR 231
UIC2_SR 214
UIC2_TR 237
USB 809
USB 2.0 Clocking 815
USB 2.0 Configuration 810
USB 2.0 Device interface 863
  FIFOs 885
  indexed registers 876
  memory map 865
  registers 866
  software interface 865
USB 2.0 host interface 816
  registers 818
USB 2.0 PHY Macro Sharing 813
USB interface 809
USB2D0_FADDR 869
USB2D0_FRAME 875
USB2D0_INCSR 878
USB2D0_INCSR0 877
USB2D0_INDEX 875
USB2D0_INMAXP 880
USB2D0_INTRIN 867
USB2D0_INTRINE 870
USB2D0_INTROUT 870
USB2D0_INTROUTE 873
USB2D0_INTRUSB 872
USB2D0_INTRUSBE 871
USB2D0_OUTCOUNT 884
USB2D0_OUTCOUNT0 884
USB2D0_OUTCSR 881
USB2D0_OUTMAXP 883
USB2D0_POWER 868
USB2D0_TSTMODE 874
USB2HMDCR0_BCNTL 858
USB2HMDCR0_BSTAT 859
USB2HMDCR0_LAT 860
USB2HMDCR0_REVID 861
USB2HSDCR0_BCNTL 862

USB2HSDCR0_REVID 863

Z

ZMII bridge
  features 683
  interface signals 683
  interfaces 683, 689
  registers 684, 689
ZMII0_FER 685
ZMII0_SMIISR 686
ZMII0_SSR 685

Revision Log

Revision Date	Version	Description
09/29/2005	1.01	Creation of preliminary book.
10/14/2005	1.02	Move processor overview to the PPC440 Processor User's Manual.
01/30/2006	1.03	Clean up document structure and formatting. Correct UICx_VCR and UICx_VR register names and address correlation. Add new DDR SDRAM section. Update Clocking section. Update DMA section. Add new SRAM section. Update CPM section. Update UART section.
02/15/2006	1.04	Remove PLB Performance Monitor. Replace Ethernet with 440GX material and update. Update GPIO, Overview, PCI, On Chip Buses, Reset, EBC, NAND Flash, and Bootstrap,
04/03/2006	1.05	Update USB chapter (update Device support and add new Host support). Add KASUMI algorithm chapter. Add Security Function chapter. Implement numerous updates from review of version 1.04.
04/21/2006	1.06	Misc. updates
05/26/2006	1.07	Misc. updates
06/09/2006	1.08	Update USB Host Controller register field data.
07/24/2006	1.09	Miscellaneous corrections.
11/01/2006	1.10	Add comments to EMACx_ISR[ORE] and EMACx_ISR[IRE] Correct references to Bootstrap options in SDR0_SDSTP0:3 registers. Remove reference to two Mentor Graphics USB documents. Update description and field comments for EMACx_STACR register. Remove Confidential status.
02/12/2007	1.11	Change DDR0_CFGADDR register from R/W to W Correct erroneous references from GPIOx_OSR registers to GPIOx_ISR registers. Correct DDR0_nn register offset from decimal to hex.Add additional information concerning booting from NAND Flash. Correct critical and non-critical settings for MSR[EE] and MSR[CE] bits. Correct PCI ID value. Correct PCIC0_CLS[SUB] reset value. Correct settings for GPIO38 and GPIO39 in GPIO chapter. Update settings for UCI0 bits 21 and 22. Correct offset indication for DDR registers in the register summary chapter.
04/25/2007	1.12	Move Security Function section from External Interfaces to System Operations section. Change Bootstrap options. Remove all references to 12 and 24 MHz crystals and oscillators in USB. Correct DDR register addresses. Correct multiple clocking specifications. Change PCIC0_CLS reset value. Correct PerCS0:n to PerCS0:5 wherever incorrect number (n) is shown.

Revision Date	Version	Description
11/27/2007	1.13	Misc. updates. Updates to Figure 9-13 and Figure 9-19. Corrections to GPT chapter, USB chapter, DMA chapter, UICO chapter and the NDFC chapter. Updates to Clocking chapter, table 14-10 and 14-11, updates in MAL chapter, updates in EMAC to PHY Interface Bridges - Figure 26-4 table and GPIO section. Update to Security section, added figure 17-62 and 17-63 Update EMACx - STACR [IMS, PADR] register field descriptions.
03/19/2008	1.14	Update of the Reset Values in 6.4.2.6, 6.4.2.10, 15.2.43, 15.2.44, 15.2.50, 15.2.70 and 15.2.74. Update to Figure 6-11, 6-12 and 6-16. Update to table 25-6 and 25-7. Added information to section 22.6.2 and section 22.6.3. Changes to NAND Flash chapter. Update to figure 9-12 register bit 23. Correction FIFO Size for UARTs in Serial Port Operations chapter. The correct size is 16 bytes. Updated the description of UART serial clock and divisor formulas in the Serial Port Operations chapter. Updated UARTx_FCR[RFTL]. Replaced NS16750 with NS16550. Updated MAL master priority register fields in Table 2-3. Numerous update to the Nand Flash chapter. Update to security chapter, section 17.1.3.4, added information to last paragraph. Update to MAL section: Figures 25-5 and 25-7. Removed references to unsupported RTBI and TBI modes. Removed EMAC GPCS registers used by RTBI and TBI modes.
09/22/2008	1.15	Doc issue 497: Updated SDR0_USB2PHY0CR[XOREF] description, Figure 32-5 and Figure 32-6. Doc issue 502: Replaced 40ns and 120ns with 40ps and 120ps. Change affects DDR0_09[WDSB], DDR0_14[DDRBYP], DDR0_20[DDDB0], DDR0_22[DQS0SB] bit fields. Doc issue 511: Removed empty figure. Affected Figure 9-3. UIC1 Status Register (UIC1_SR). Doc issue 514: Corrected UIC register bit numbers in section 13.3.1 Compare Timer Interrupt. Doc issue 522: Updated section 27.2.9 Packet Reject. Doc issue 541: Corrected DDR0_07[TFAW] bit field numbering typo. Doc Issue 555: Corrected MAL0_ESR[CIDT] description error. Doc issue 569: Updated NAND flash feature list and corrected NAND flash interface recommendation for 16-bit NAND flash. Doc issue 578: Correct errors in EMACx_TMR0[TFAE] and EMACx_RMR[RFAF] field description Merged the PPC440EPx and PPC440GRx User's Manual. Placed comments throughout the user's manual indicating PPC440EPx only supported features. Doc issue 585: Added recommendation for Primary A and B Divisors. This update affects SDR0_SDSTP0[PRBDV0], CPR0_PRIMBD0[PRADV0] and CPR0_PRIMBD0[PRBDV0]. Added recommendation for MAL divisor. This update affects CPR0_MALD[MALDV0]. Updated Clock Ratio Listing tables in Clocking chapter. Added alignment exception to load/store double instructions.